TNT Input Engine

A Godot 3.x Unified Input System

Contact: gregbug@gmail.com

A cosa serve? Perché è stato creato?

TNT input engine è un “addon” realizzato per il motore di gioco Godot 3.x.

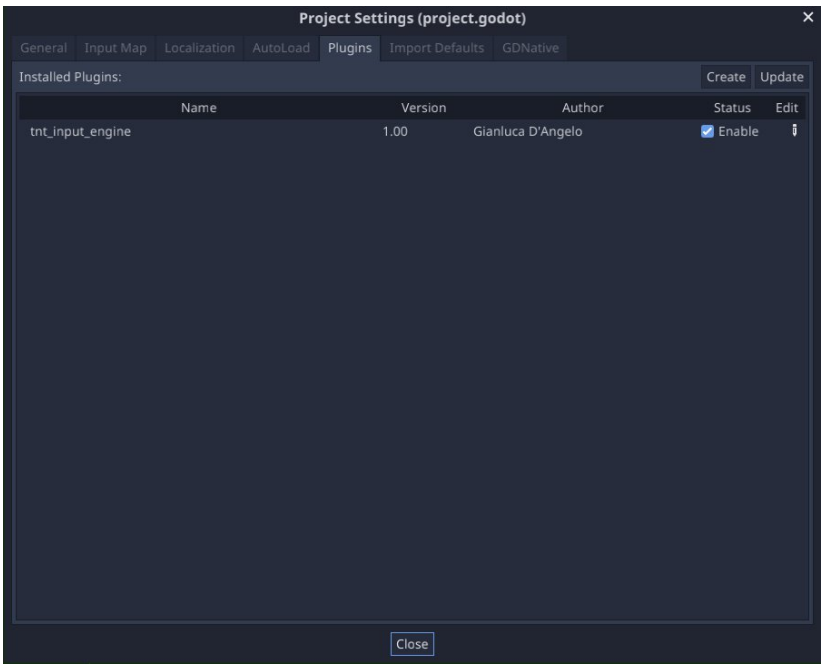
Il motivo principale per il quale è stato realizzato è per semplificare ed unificare la scrittura di codice di controllo di input permettendo di scrivere il codice una volta sola e gestire simultaneamente l’input da tastiera da joystick virtuale touchscreen e joystick reale utilizzando le API standard di Godot.

È stato pensato fin dall'inizio per non impattare sulle performance ed infatti il sistema utilizza l’input engine di Godot senza aggiungere nessun sovraccarico; l’unico codice aggiuntivo è per la gestione del touchscreen per il joypad virtuale e dove il codice è davvero leggero e veloce le gestione dell’input è invece demandata all’engine stesso.

Per rendere la vita del programmatore più semplice è stato inoltre aggiunto un sistema di mappatura semplificato e automatizzato per tutti quei dispositivi di input che non vengono riconosciuti in modo nativo da Godot.

Installazione

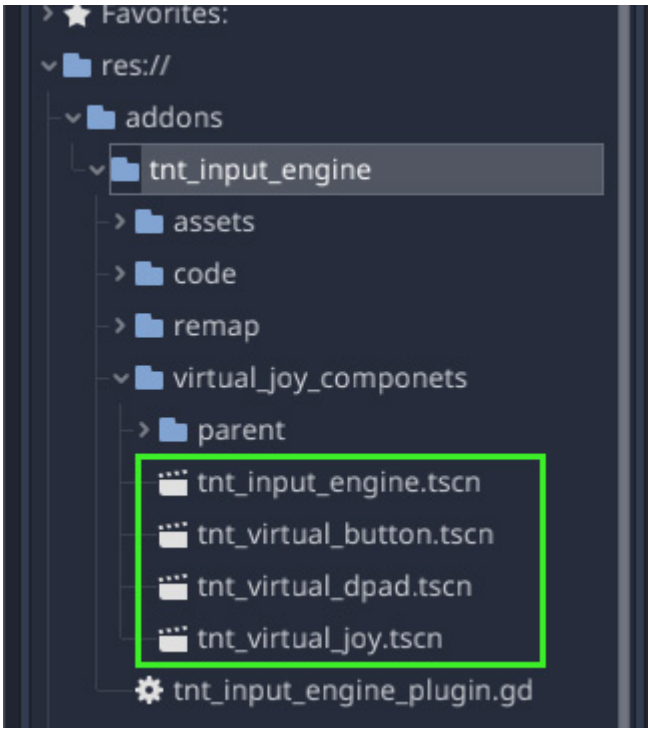
L’installazione del plugin segue le regole di Godot; copiare od importare il plugin sotto la cartella addons del vostro progetto ed attivatelo dal menu Godot Project->Project setting->Plugins ed abilitate “tnt_inpuit_engine”.



Ok, ora potete utilizzare il plugin!

PS: Durante lo sviluppo su PC desktop ricordarsi di abilitare emulare il touchscreen tramite mouse. (Godot Project->Project setting->Input devices->Pointing e abilitare "emulate touch from mouse").

Ora se vi spostate sotto la cartella addons osservate il contenuto della cartella "virtual_joy_components" qui troverete quattro elementi saranno loro che vi serviranno per gestire l'input.



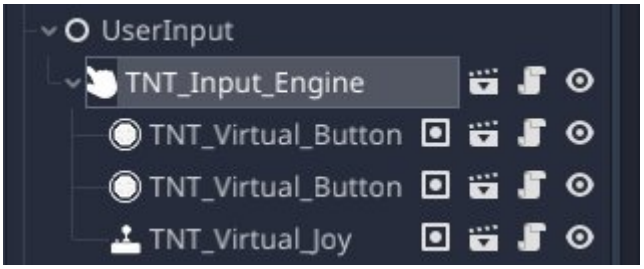
tnt_input_engine.tscn è il nodo genitore che servirà a gestire le propietà globali del sistema.

tnt_virtual_button.tscn è il nodo che gestisce i bottoni di input.

tnt_virtual_dpad.tscn è il nodo che gestisce il "directional" pad digitale. (virtuale e reale)

tnt_virtual_joy.tscn è il nodo che gestisce il joystick analogico (virtuale e reale)

La composizione di questi nodi vi permetterà quindi di comporre il vostro joypad touchscreen che si comporterà esattamente come il joypad reale quando questo non è presente condividendo lo stesso codice di controllo (codice standard di Godot, non dovrete quindi imparare nuove API)



È importante notare che i nodi

tnt_virtual_button, tnt_virtual_dpad, tnt_virtual_joy

DEVONO NECESSARIAMENTE ESSERE "FIGLI" di tnt_input_engine.

Ora per velocizzare questa specie di corso veloce aprite la scena Demo_1 sotto la cartella demo e avviatela.

Se tutto va bene potrete muovere il personaggio con la tastiera, con il dpad virtuale touchscreen o con il joypad! E la parte bella è che il codice di controllo è sempre questo:

```
func _physics_process(_delta: float) -> void:
	if  Input.is_action_pressed(TNTInputEngine.ACTION_JOY_LEFT):
			player_anim.flip_h = true
			player_anim.play("right")
			player_xy.x = -1
	elif Input.is_action_pressed(TNTInputEngine.ACTION_JOY_RIGHT):
			player_anim.flip_h = false
			player_anim.play("right")
			player_xy.x = 1
	elif Input.is_action_just_released(TNTInputEngine.ACTION_JOY_LEFT) ||
Input.is_action_just_released(TNTInputEngine.ACTION_JOY_RIGHT):
			player_xy.x = 0
			player_anim.stop()
			player_anim.frame = 1
	if Input.is_action_pressed(TNTInputEngine.ACTION_JOY_UP):
			player_xy.y = -1
			if player_xy.x == 0:
				player_anim.play("up")
	elif Input.is_action_pressed(TNTInputEngine.ACTION_JOY_DOWN):
			player_xy.y = 1
			if player_xy.x == 0:
				player_anim.play("down")
```

Quello che vi consiglio è di provare e modificare demo_1 e demo_2.

Seguirà ora la descrizione delle classe globale TNTInputEngine e i nodi da utilizzare per gestire l'input.

TNTInputEngine Class

TNTInputEngine è la classe globale introdotta dal plugin e che vive quindi durante tutta l'esecuzione del vostro progetto.

In questa classe vengono definite costanti e funzioni di utilità per l'uso con il plugin.

Il codice sogente della classe "TNTInputEngine" è contenuto nel file
res://addons/tnt_input_engine/code/global_class/tnt_input_engine_global.gd

Come in tutto il codice sorgente dell'addon tutte le variabili, costanti, funzioni che cominciano con "_" sono da considerarsi private e quindi da non utilizzare al di fuori delle classi dell'addon.

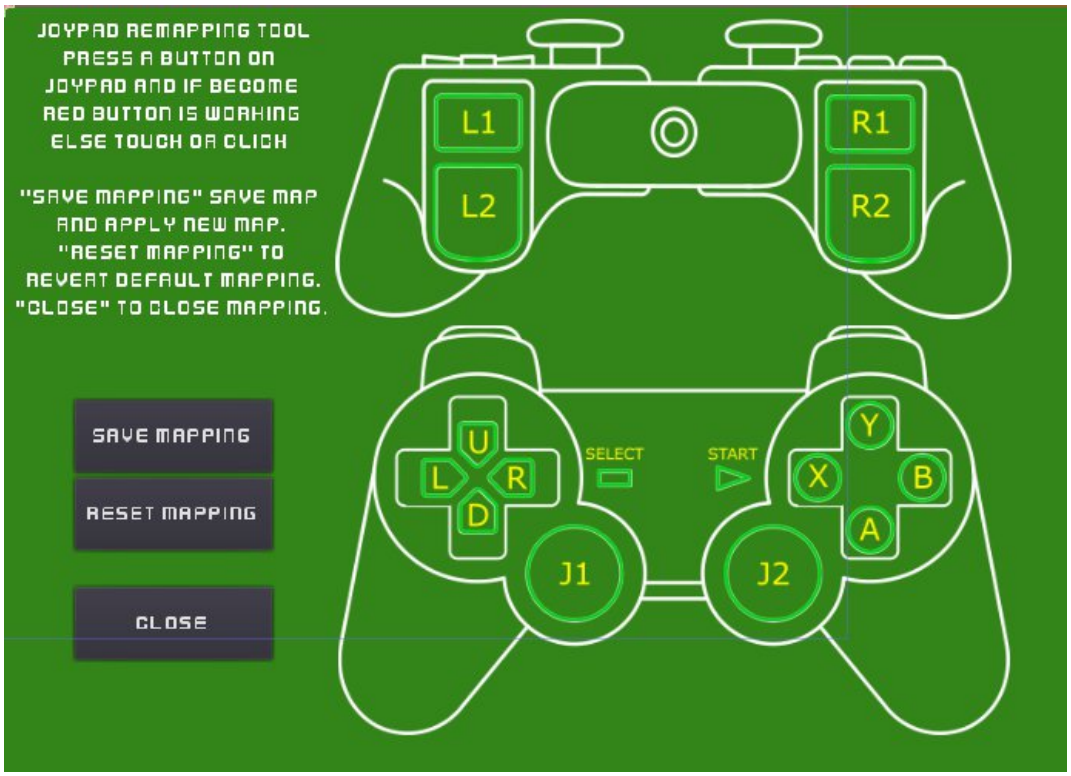
Il principio base per l'utilizzo corretto di questo addon è che se lo utilizzate sarebbe meglio non utilizzare l'inputmap di godot ma utilizzare per questo scopo solo l'addon.

Dopo questo piccolo preambolo analizziamo le funzionalità della classe TNTInputEngine:

func open_hw_joy_remap(device_id: int) -> void:

device_id = id del joypad da rimappare.

Apre la scena (si adatta automaticamente al formato landscape o portrait) che permette di rimappare in modo visuale il joypad specificato da device_id:



Terminata la mappatura l'utente può scegliere se salvarla per poi essere riutilizzata in qualsiasi momento. (per maggiori info vedere Demo_1 scena HW_JoyInfo codice res://demo/shared/HW_JoyInfo.gd funzione update_joypad_info())

func add_key_binding(key_scancode: int, action_name: String) -> void:

Permette di collegare l'input da tastiera al joypad virtuale e al joypad reale.

key_scancode: è lo scancode di Godot che rappresenta il tasto che si desidera utilizzare ad esempio KEY_UP, KEY_1, KEY_A

action_name: è l'azione che si vuole associare al tasto. E' una stringa definita dall'utente oppure una stringa predefinita da TNTInputEgine (vedere Demo_1 codice res://demo/scenes/code/demo_1.gd):

```
const ACTION_JOY_A: String = "joy_a"
const ACTION_JOY_B: String = "joy_b"
const ACTION_JOY_Y: String = "joy_y"
const ACTION_JOY_X: String = "joy_x"
const ACTION_JOY_START: String = "joy_start"
const ACTION_JOY_SELECT: String = "joy_select"
const ACTION_JOY_L2: String = "joy_l2"
const ACTION_JOY_R2: String = "joy_r2"
const ACTION_JOY_L1: String = "joy_l1"
const ACTION_JOY_R1: String = "joy_r1"
const ACTION_JOY_UP: String = "joy_up"
const ACTION_JOY_LEFT: String = "joy_left"
const ACTION_JOY_DOWN: String = "joy_down"
const ACTION_JOY_RIGHT: String = "joy_right"
const ACTION_LEFTAXIS_UP: String = "axis_lefty-"
const ACTION_LEFTAXIS_RIGHT: String = "axis_leftx+"
const ACTION_LEFTAXIS_DOWN: String = "axis_lefty+"
const ACTION_LEFTAXIS_LEFT: String = "axis_leftx-"
const ACTION_RIGHTAXIS_UP: String = "axis_righty-"
const ACTION_RIGHTAXIS_RIGHT: String = "axis_rightx+"
const ACTION_RIGHTAXIS_DOWN: String = "axis_righty+"
const ACTION_RIGHTAXIS_LEFT: String = "axis_rightx-"
```

Tabella 1

func add_joy_binding(button: int, action: Array = ["", "", "", ""], dead_zone: float = 0.5) -> void:

Permette di collegare via codice componenti del virtual joypad e del joypad reale.

button: joy.DPAD, joy.JOY_LEFT, joy.JOY_RIGHT, joy.BTN_A, joy.BTN_B, joy.BTN_X, joy.BTN_Y, joy.SELECT, joy.START, joy.L1, joy.L2, joy.R1, joy.R2

action: azione (definita dall'utente o tabella 1) da collegare al componente (per dpad, joy_left e joy_right le azioni sono 4) se non definita viene utilizzato il valore predefinito.

joy.DPAD: ACTION_JOY_UP, ACTION_JOY_RIGHT, ACTION_JOY_DOWN, ACTION_JOY_LEFT

joy.JOY_LEFT: ACTION_LEFTAXIS_UP, ACTION_LEFTAXIS_RIGHT, ACTION_LEFTAXIS_DOWN, ACTION_LEFTAXIS_LEFT

joy.JOY_RIGHT: ACTION_RIGHTAXIS_UP, ACTION_RIGHTAXIS_RIGHT, ACTION_RIGHTAXIS_DOWN, ACTION_RIGHTAXIS_LEFT

E cosi via...

dead_zone: valore al disotto del quale il joypad non si muove. (0.5 se non definito)

func delete_dbfile(joy_guid: String) -> int:

Cancella se esistente il file di mappatura joypad con GUID **joy_guid**

Rilascia OK se cancellazione è andata a buon fine.

func user_dbfile_joy_exist(joy_guid: String) -> bool:

Riascia true se il file di mappatura del joypad **joy_guid** esiste altrimenti rilascia false

func load_and_apply_joy_db(device_id: int) -> int:

Legge ed applica la mappatura del joypad con id **device_id** se esiste.

Rilascia OK se tutto regolare altrimenti rilascia ERR_FILE_NOT_FOUND.

func set_active_joy_id(id: int) -> void:

Imposta il joypad attivo con indice **id**

func get_active_joy_id() -> int:

Rilascia l'id del joypad attualmente attivo

func get_connected_devices_count() -> int:

Rilascia il numero dei joypad attualmente collegati

func get_device_id(device_index: int) -> int:

Rilascia l'id del joypad con indice **device_index**

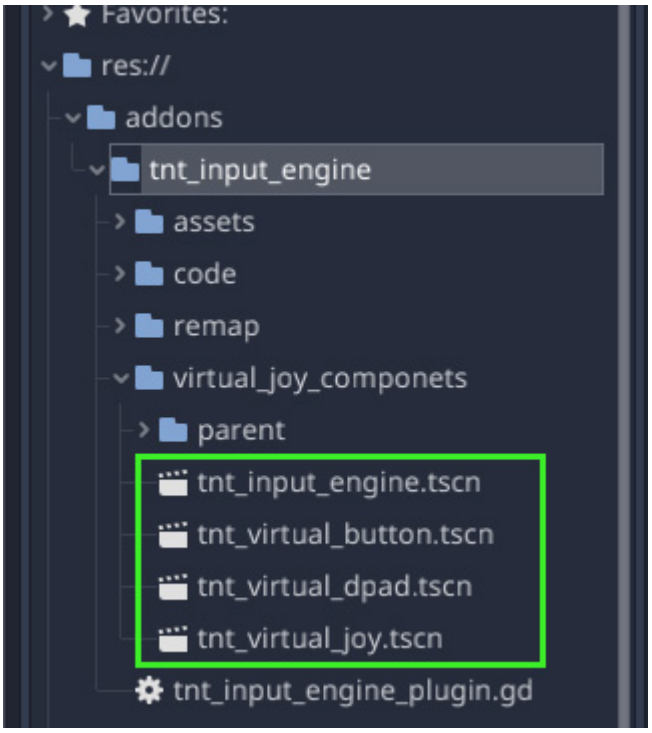
func get_device_name(device_id: int) -> String:

Riascia il nome del joypad con id **device_id**

func get_device_GUID(device_id: int) -> String:

Rilascia la GUID del joypad con id **device_id**

Nodi TNTInputEngine



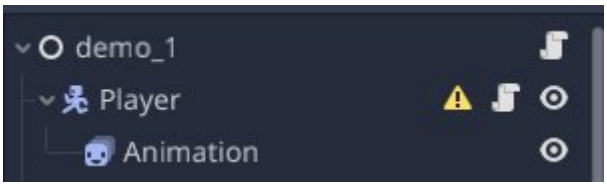
Come già detto in precedenza i nodi che compongono tutto il sistema sono questi quattro e sono nodi visuali essi permettono di costruire un sistema di input dinamico ed efficiente ovviamente troverete questi nodi al di sotto della cartella Addons->tnt_input_engine->virtual_joy_components, il componente più importante è il primo tnt_input_engine che è il nodo padre e che controllerà tutti i sotto nodi.

Nel prossimo paragrafo spiegherò come inizializzare e costruire il primo joystick virtuale ovviamente vi ricordo che è fortemente consigliato studiare e controllare i demo allegati al plugin nello specifico demo_1 e demo_2.

Per l'inserimento dei nodi nel vostro progetto basta trascinarli dalla cartella a Addons all'interno della vostra scena.

Come iniziare

Per iniziare apriamo il demo_1 ora se osserviamo al lato sinistro ovvero nella gerarchia della scena troveremo la seguente situazione:



Clicchiamo nel simbolo del codice del nodo demo_1 e osserviamo la seguente porzione di codice:

```
func _ready() -> void:
	################################################
	# adding keyboard actions binding via code.... #
	################################################
	# FOR SCENE 1
	TNTInputEngine.add_key_binding(KEY_UP, TNTInputEngine.ACTION_JOY_UP)
	TNTInputEngine.add_key_binding(KEY_DOWN, TNTInputEngine.ACTION_JOY_DOWN)
	TNTInputEngine.add_key_binding(KEY_LEFT, TNTInputEngine.ACTION_JOY_LEFT)
	TNTInputEngine.add_key_binding(KEY_RIGHT, TNTInputEngine.ACTION_JOY_RIGHT)
	# FOR SCENE 2
	TNTInputEngine.add_key_binding(KEY_1, TNTInputEngine.ACTION_JOY_A)
	TNTInputEngine.add_key_binding(KEY_2, TNTInputEngine.ACTION_JOY_B)
	TNTInputEngine.add_key_binding(KEY_UP, TNTInputEngine.ACTION_LEFTAXIS_UP)
	TNTInputEngine.add_key_binding(KEY_DOWN, TNTInputEngine.ACTION_LEFTAXIS_DOWN)
	TNTInputEngine.add_key_binding(KEY_LEFT, TNTInputEngine.ACTION_LEFTAXIS_LEFT)
	TNTInputEngine.add_key_binding(KEY_RIGHT, TNTInputEngine.ACTION_LEFTAXIS_RIGHT)

	#######################################################
	# adding VPAD/JOYPAD for scene 2 binding via code.... #
	#######################################################
	TNTInputEngine.add_joy_binding(TNTInputEngine.joy.JOY_LEFT)
	TNTInputEngine.add_joy_binding(TNTInputEngine.joy.BTN_A)
	TNTInputEngine.add_joy_binding(TNTInputEngine.joy.BTN_B)
```

In questa prima parte del codice andiamo a legare l'input da tastiera con l'input del joystick virtuale e del joystick reale come già detto è preferibile inserire tutto ciò tramite codice ma in realtà è anche possibile farlo tramite l'interfaccia utente di Godot.

La seconda parte del codice andiamo a collegare per la scena 2 un joypad analogico sinistro virtuale e due bottoni (bottone a e il bottone B) siccome non vengono specificati altri parametri le azioni saranno quelle standard definite dalla libreria TNT come specificato nella tabella 1.

```
TNTInputEngine.add_key_binding(KEY_UP, TNTInputEngine.ACTION_JOY_UP)
```

Con questa linea di codice quindi leghiamo al tasto “su” l’azione TNTInputEngine.ACTION_JOY_UP e che sarà emesso ogni qualvolta si preme il tasto “su”.

Primo Joypad Virtuale

Apriamo Demo_2 tralasciamo gli altri nodi ed osserviamo attentamente quelli sotto il nodo user input.

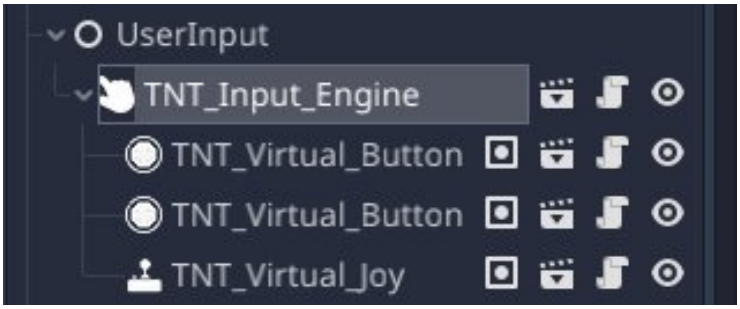
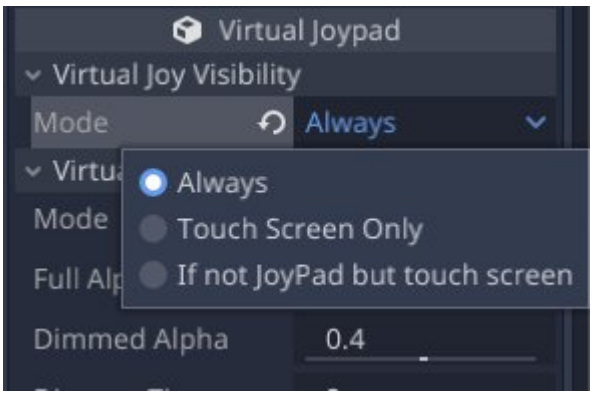


Figura 1.

Per questione di ordine consiglio di inserire sempre un nodo primario chiamato “userinput” al di sotto del quale andremo a costruire il nostro joypad. Per inserire il nodo TNT_input_engine basterà trascinarlo dalla cartella addon al di sotto del nodo user input. Questo nodo è il nodo principale del nostro joypad e dovrà essere necessariamente genitore di tutti gli altri sottonodi e ci permetterà di controllare il comportamento di tutto il Joypad; di seguito analizziamo le proprietà principali:



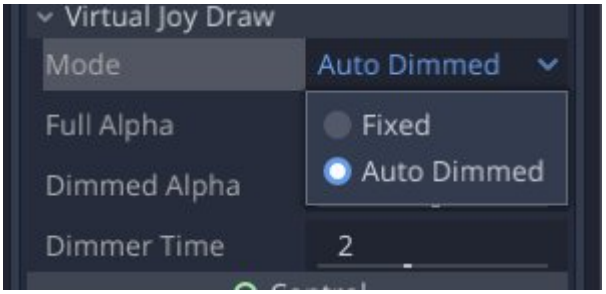
Virtual Joy Visibility:

Always: il joypad virtuale è SEMPRE visibile.

Touch Screen Only: il joypad virtuale è visibile SOLO se il dispositivo ha il touch screen.

If Not joyPad but touch screen: se è presente un joypad reale quello virtuale non sarà visibile, se il joypad reale non è presente il joypad virtuale sarà visibile solo se è presente il touch screen.

PS: Nota bene tutti i metodi di input saranno sempre e comunque funzionanti se presenti (eccetto il joypad virtuale che è funzionante solo se visibile).



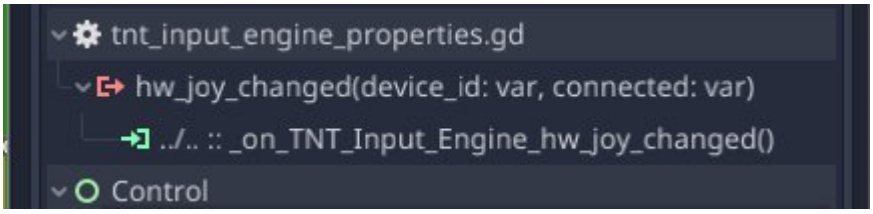
Virtual Joy Draw:

Fixed: I componenti del joypad virtuale saranno sempre disegnati con un Alpha uguale a 1 quindi come originariamente disegnati.

Auto Dimmed: i conponenti del joypad virtuale saranno disegnati con una trasparenza Alpha uguale a *Full Alpha* e

dopo il tempo (secondi) definito in *dimmer time* con un Alpha pari al valore di *Dimmed Alpha*; simulando quindi una trasparenza a tempo.

Eventi del Nodo TNT_input_engine:



Nel nodo TNT_input_engine è definito un evento molto importante che viene eseguito ogni qualvolta si inserisce o disinserisce un joypad hardware nel pc o nel dispositivo.

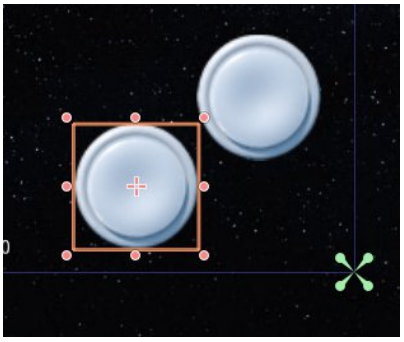
Questo ci permette di capire ovviamente quando si inserisce o collega un joypad e prendere eventualmente le decisioni appropriate.

L'evento porta con sé due parametri molto importanti il *device ID* che contiene l'ID del joypad e l'informazione *connected* per capire se si è inserito (true) ho disinserito (false) il joypad.

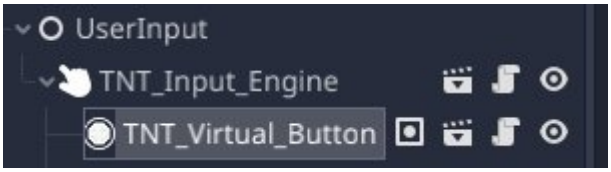
Vedere Demo_1 per un esempio concreto di utilizzo.

Ora che abbiamo definito il comportamento generale del nostro joypad virtuale andiamo ad aggiungere dei bottoni e un joypad analogico per il movimento dell'astronave come illustrato nel demo_2.

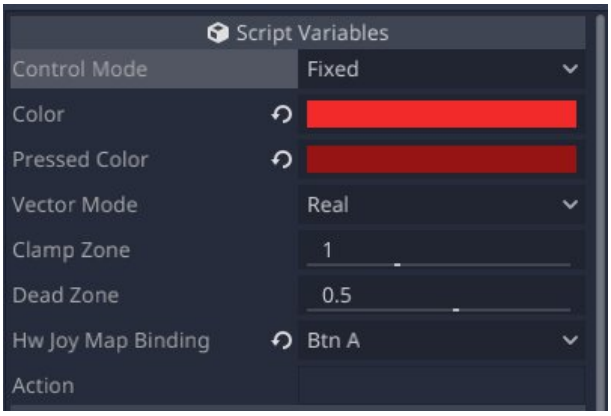
TNT_Virtual_Button



Il nodo *tnt_virtual_button* è stato creato per gestire i “bottoni” o “pulsanti” sia della tastiera, virtual pad o joypad reale. Se ne possono creare quanti ne vogliamo ovviamente se mappati a joypad reali si è limitati dal numero reale di pulsanti del joypad. Ad ognuno di essi si potrà “legare” tramite binding sia la tastiera che il joypad. Ogni bottone può avere un comportamento e un colore autonomo. Per rilevarne la pressione si seguirà l’input standard di Godot.



Come già detto più volte il nodo *tnt_virtual_button* dovrà essere inserito come figlio al di sotto del nodo TNT_input_engine altrimenti non funzionerà.



Le propietà specifiche del *tnt_virtual_button* sono:

Control Mode:

Fixed: Il bottone o componente sarà sempre fisso dove è stato posizionato in fase di progettazione.

Dynamic: Il bottone o componente si sposterà nel punto di pressione del touch.

Follow: Il Bottone o componente segue il movimento di pressione del touch.

Color: Specifica il colore predominante del Bottone o componente quando non premuto.

Pressed Color: Specifica il colore predominante del Bottone o componente quando premuto.

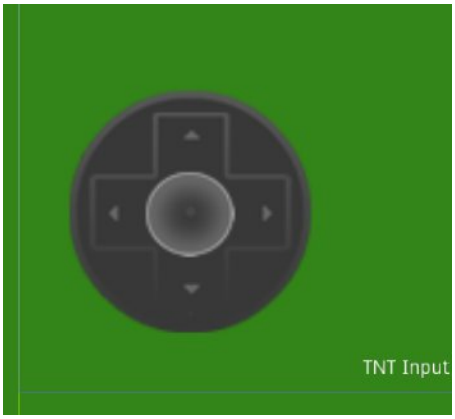
Vector Mode, Clamp zone, Dead zone: Non sono di rilevanza per il componente *tnt_virtual_button.*

Hw Joy Map Binding: Se diverso da “none” viene utilizzato per “mappare” il bottone corrispondente al joypad reale. I valori possibili sono: NONE, BTN_A, BTN_B, BTN_X, BTN_Y, SELECT, START, L1, L2, R1, R2.

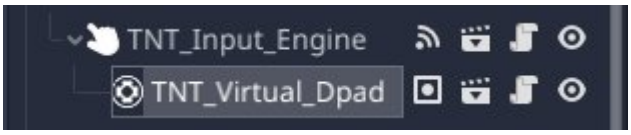
Action: E’ una stringa che rappresenta l’azione che verrà emessa alla pressione del bottone o del componente e che dovrà essere intercettata dal codice per eseguire l’azione. Se la stringa viene lasciata vuota verrà utilizzata l’azione standard TNTInputEngine come da tabella 1. Ad esempio se **Hw Joy Map Binding** è assegnato il valore BTN_A e **Action** è vuota verrà assegnato il valore:
`TNTInputEngine.ACTION_JOY_A`

Quindi per rilevarne poi la pressione baserà: `if Input.is_action_pressed(TNTInputEngine.ACTION_JOY_A)` Non importa se proviene da tastiera, joystick virtuale o joystick reale l’evento sarà rilevato. (Per l'inserimento tramite tastiera ricordo che dovrà essere necessariamente abilitato tramite codice come già visto in precedenza `TNTInputEngine.add_key_binding(KEY_A, TNTInputEngine.ACTION_JOY_A)`)

TNT_VIRTUAL_DPAD



Il nodo *TNT_virtual_dpad* è stato creato per simulare il pad direzionale digitale del joypad; può essere impostato per gestire movimenti solo orizzontale, solo verticali o ambedue contemporaneamente. Anche qui se ne possono creare quanti ne vogliamo ovviamente se mappati a joypad reali si è limitati dal numero reale di DPAD del joypad. Ad ognuno di essi si potrà “legare” tramite binding sia la tastiera che il joypad. Ogni bottone può avere un comportamento e un colore autonomo. Per rilevarne la pressione si seguirà l’input standard di Godot.



Come già detto più volte il nodo *TNT_virtual_dpad* dovrà essere inserito come figlio al di sotto del nodo TNT_input_engine altrimenti non funzionerà.

Le propietà specifiche del *tnt_virtual_dpad (oltre a quelle comuni al tnt_virtual_button)* sono:

Dpad Directions:

All Direcrtions: il dpad si potrà muovere in tutte le direzioni.

Left Right: il dpad si potrà muovere solo in orizzontale.

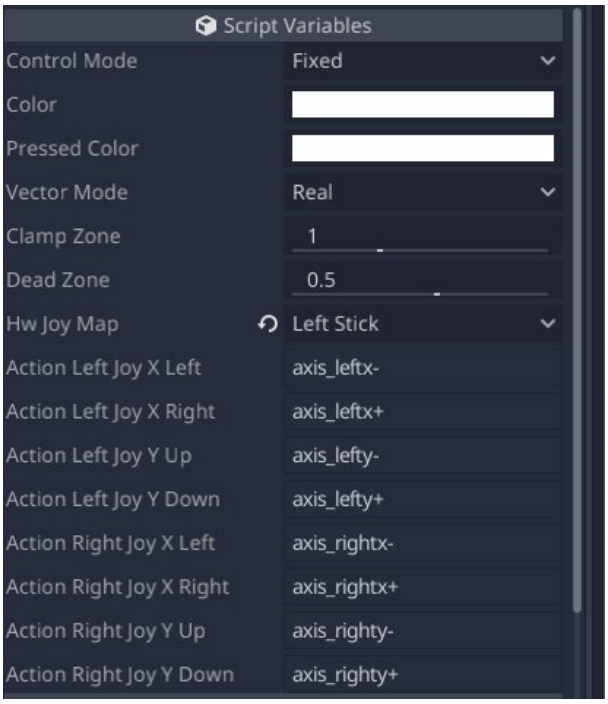
Up Down: il dpad si potrà muovere solo in verticale.

Action up, Action Right, Action down, Action Up: E' una stringa che rappresenta l'azione che verrà emessa alla pressione del bottone o del componente e che dovrà essere intercettata dal codice per eseguire l'azione. Se la stringa viene lasciata vuota verrà utilizzata l'azione standard TNTInputEngine come da tabella 1. Di default sono impostate le stringhe di azioni standard.

TNT_Virtual_Joy

I nodo *TNT_virtual_joy* è stato creato per simulare il joystick analogico del joypad; Anche qui se ne possono creare quanti ne vogliamo ovviamente se mappati a joypad reali si è limitati dal numero reale di joystick del joypad (tipicamente 2). Ad ognuno di essi si potrà "legare" tramite binding sia la tastiera che il joypad. Ogni bottone può avere un comportamento e un colore autonomo. Per rilevarne il movimento si seguirà l'input standard di Godot.

Come già detto più volte il nodo *TNT_virtual_joy* dovrà essere inserito come figlio al di sotto del nodo TNT_input_engine altrimenti non funzionerà.



Le propietà specifiche del *tnt_virtual_joy (oltre a quelle comuni al tnt_virtual_button e tnt_virtual_dpad)* sono:

Vector Mode:

Real: vettore valore reale da 0 a 1

Normalized: vettore normalizzato

Clamp Zone: 0..2 distanza massima possibile dal centro del joystick.

Dead Zone: valore al disotto del quale non vengono rilevati movimenti.

Hw joy Map: Joystick con cui avviene la mappatura (nessuno, sinistro o destro).

Action left/right: E' una stringa che rappresenta l'azione che verrà emessa al movimento del joystick e che dovrà essere intercettata dal codice per eseguire l'azione. Se la stringa viene lasciata vuota verrà utilizzata l'azione standard TNTInputEngine come da tabella 1. Di default sono impostate le stringhe di azioni standard.

PS: Quando i componenti del joypad virtuale non sono visibili nessun codice viene eseguito in modo da non impattare in nessun modo sulle performances.

Dopo questa breve introduzione quello che mi sento di consigliare è di verificare e approfondire il demo 1 e 2 provando a sperimentare modificando i parametri dei nodi.

Spero che questo addon possa tornarvi utile come lo è stato per me se vi piace lo utilizzate e volete aiutarmi potete supportarmi nello sviluppo nella correzione e ottimizzazione contattandomi direttamente alla mia e-mail oppure se vi va potete offrirmi una birra tramite una piccola donazione PayPal direttamente sulla mia e-mail. (Gianluca D'Angelo gregbug@gmail.com)

Siete liberi di utilizzarlo per qualsiasi vostro progetto sia commerciale che gratuito.

Grazie, Gianluca D'Angelo (GregBUG)

L'Aquila, Italy.