InputControllerFx

Table des matières

I – Définition

InputControllerFx est un module conçut pour la gestion des différentes entrées de l'utilisateur (Clavier, Souris, et Manette). L'objectif de ce module est de permettre aux développeurs de pouvoir facilement géré les entrées de ses clients sans trop se gèner. Concernant l'utilisation de ce dernier, le développeur doit être précis sur les touches dont-il souhaite prendre en charge dans son jeu.
Une mauvaise utilisation peut conduire à des bugs. Notez que ce module prend également en charge les changements de touches dynamiques (*keybindings*).
NB : Ce module est compatible à un jeu 2D, 3D et est sauvegardable. Avant de continuer, le module ne supporte pas encore les écrans tactiles.

II – Les propriétés disponibles

+ int PlayerIndex : Quel joueur voulez-vous écouter ? Notez que cette option n'est destiné qu'aux manettes.

+ bool DisableMouseMotion = true : Contrôle l'activation ainsi que la désactivation de la prise en charge des mouvements de la souris lors d'une opération de changement de touche (*keybindings*).

+ Array Inputs : Tableau de dictionnaires contenant toutes les différentes configurations sur chaque touche prise en charge par le développeur. Les dictionnaires issus de ce tableau supportent les clés suivantes :

» **int controller = 0** : Quel contrôleur voulez- vous écouter ? Les valeurs possibles sont :
-> **InputControllerFx.KEYBOARD ou 0** : Ecoute le clavier de l'ordinateur.
-> **InputControllerFx.MOUSE ou 1** : Ecoute la souris de l'ordinateur.
-> **InputControllerFx.JOYSTICK ou 2** : Ecoute la manette connectée à l'ordinateur.
-> **InputControllerFx.INPUTMAP ou 3** : Ecoute la carte des entrées définit au sein de Godot.
-> **InputControllerFx.ALL ou 4** : Ecoute l'entrée de n'importe quel contrôleur.

» **String key** : Quel est le nom de la touche du contrôleur à mettre en écoute ? A ce niveau, il y a certaines restrictions que vous devez respecter. Si le contrôleur est le clavier, vous êtes plié de mettre le nom correspondant à la touche recherchée tout en respectant ce que renvoie Godot lorsqu'on appuie sur une telle touche. Si le contrôleur est la souris, en cas de bouton, il doit suivre la nomenclature **Mouse+IndexDeLaToucheVoulue**.
Exemple : *Mouse2, Mouse1, ...MouseN*.
En cas d'axe, le ciblage des défilements doit se faire avec les mots clés suivants : *-ScrollX, +ScrollX, -ScrollY, +ScrollY* et celui des mouvements avec -MouseX, +MouseX, -MouseY, +MouseY. Si le contrôleur est la manette, en cas de bouton, il doit suivre la nomenclature **Joy+IndexDeLaToucheVoulue**.
Exemple : *Joy0, Joy1, ...JoyN*.
En cas d'axe, suivre la nomenclature **Signe+Axis+IndexDeLaxeVoulu**.
Exemple : *-Axis2, +Axis0, ...-/+AxisN*.

Si vous écoutez la carte d'une ou de plusieurs entrée(s) prédéfinit dans les configurations de votre projet, vous devez renseigner l'identificateur de la carte à ciblée.

» **bool any = false** : Voulez-vous écouter toutes les touches du contrôleur actuel ?

» **bool translate = false** : Voulez-vous traduire la touche actuelle en terme plus claire pour les futures utilisateurs de votre application ? L'activation de cette option essaye de faire une traduction objective de la touche détectée tout en s'adaptant au contrôleur détecté. Cela ne marche pas à tous les fois. Cette option n'intervient qu'en mode *keybindings*. Notez qu'elle n'est pas activée lorsque le contrôleur utilise la carte des entrées définit au sein de Godot.

» **bool changed = false** : Voulez-vous changer la touche actuelle ? Cette option est très utilisé dans l'implémentation d'un système de *keybindings*.

» **float strength = 0.0** : Contient la valeur d'axe maximale que renvoye un bouton. Cette option devient utile lorsqu'au cours de l'exécution du jeu, une touche d'axe a été remplacée par un bouton au cours d'un processus de changement de touche. Les valeurs possibles de cette clé sont dans l'intervalle [-1.0 ; 1.0]. Notez que cette valeur n'est renvoyée que si et seulement si la touche renseignée au niveau de la clé *key* est un bouton.

» **bool negative = false** : Quel sera le signe de la valeur de la touche ou de l'axe détecté(s) ? Par défaut, la valeur retournée est strictement positive ou égale à 0.0.

» **bool loop = false** : Souhaitez-vous mettre en écoute continue la touche à détectée ? A ce niveau, tant que celle-ci sera maintenue enfoncée, elle sera toujours détectée.

» **bool timeout = 0.0** : Quel est le délai avant chaque détection. Lorsque l'option *loop* est activée, on obtient des intervalles réguliers de détection. Dans le cas contraire, l'utilisateur est plié d'effectuer un appuie long pour déclencher l'action prévue à cet effet.

» **int type = 1** : Quel type de donnée désirez-vous renvoyée. Les valeurs possibles sont celles définient au sein de Godot. Cependant les types prises en charge sont : *TYPE_FLOAT* et *TYPE_VECTOR2*.

» **int axis = 1** : Quel axe sera considéré pour le retour des valeurs issues de la touche détectée ? Cette option prend son utilitée lorsque la valeur de la clé *type* est sur 1. Les valeurs possibles sont :
-> **MegaAssets.Axis.NONE ou 0** : Renvoie une valeur nulle ou un vecteur nulle quelque soit la détection.
-> **MegaAssets.Axis.X ou 1** : Renvoie un vecteur d'abscisse différent de zéro ou nulle.
-> **MegaAssets.Axis.Y ou 2** : Renvoie un vecteur d'ordonné différent de zéro ou nulle.

» **int count = 1** : Combien de fois la touche sera détectée avant le déclenchement de(s) action(s) prévu à son égard ?

» **int step = 1** : Quel est le pas à ajouté à chaque fois que la touche en question est détectée ? Cette option ne s'active que lorsque sa valeur et celle de *count* sont supérieur à 0.

» **bool resetlevel = true** : Voulez-vous rénitialiser le niveau d'appuie de la touche à chaque fois qu'il serait égale à sa valeur maximale ? En d'autres termes, doit-on remettre à zéro le niveau d'appuie de la touche lorsqu'elle déclenche l'action qui lui est due ?

» **int decrease = 0** : Quel est le pas à enlevé lorsque la touche détectée est relâchée ? Cette option permet de mettre en place un système de combo contre système. Elle devient utile lorsque l'on souhaite que le joueur appuie sur une certaine touche donnée de façon répété pour accomplir une action. Cette fonctionnalité fera office de décrémenteur pour ramener les éfforts de l'utilisateur à zéro, si ce dernier décide d'abandonner. Cette option ne s'active que lorsque sa valeur et celle de la clé *frequence* sont supérieur à zéro.

» **float frequence = 0.03** : Quel est le temps mort avant chaque décrémentation. Cette option ne s'active que lorsque sa valeur et celle de *decrease* sont supérieur à zéro.

» **Array keydown** : Que se passera t-il lorsqu'on appuyera sur la touche en question ? Cette clé contient tous les configurations relatives à un ou plusieurs flux d'exécution(s). L'utilisation de cette clé est déjà décrite au niveau des bases du framework. Précisement le sujet portant sur l'utilisation de la propriété EventsBindings (la section des actions d'un événement).

» **Array keyup** : Que se passera t-il lorsqu'on relâchera la touche en question ? Notez que cette dernière possède les mêmes propriétés que la clé *keydown* et s'utilise de la même manière que lui.

+ PoolStringArray InputsMap : Contient la liste complète des cartes des entrées pré-configurées dans les configurations du projet en cours de réalisation. Cette liste est également solicitée lorsque le module passe en mode *keybindings*.

+ Array ExternalInputs : Tableau de NodePath ou de String pointant tous vers les différents instances de ce module en vue d'établir une écoute générale.
NB : Les répétitions au niveau des touches ne sont pas tolérées.

III – Les méthodes disponibles

+ void keyboard_activation (activation, delay = 0.0) : Contrôle l'état du clavier en terme de permition.
» **bool activation** : Contient l'état du clavier.
» **float delay** : Quel est le temps mort avant le changement d'état ?

+ **void mouse_activation (activation, delay = 0.0) :** Contrôle l'état de la souris en terme de permition.
  » **bool activation** : Contient l'état de la souris.
  » **float delay** : Quel est le temps mort avant le changement d'état ?

+ **void joystick_activation (activation, delay = 0.0) :** Contrôle l'état de la manette en terme de permition.
  » **bool activation** : Contient l'état de la manette.
  » **float delay** : Quel est le temps mort avant le changement d'état ?

+ **bool is_keyboard_enabled () :** Le clavier est-il en état d'envoyer des signaux pour déclencher des actions ?

+ **bool is_mouse_enabled () :** La souris est-elle en état d'envoyer des signaux pour déclencher des actions ?

+ **bool is_joystick_enabled () :** La manette est-elle en état d'envoyer des signaux pour déclencher des actions ?

+ **int get_detected_key () :** Renvoie l'index de position de la touche qui a été détectée suite à un appuie ou un relâchement.

+ **PoolStringArray | String get_translation_of (id = null) :** Renvoie le(s) nom(s) réel(s) de(s) touche(s) au sein du paramètre *id* en fonction du contrôleur détecté. Par défaut, la traduction se fera sur la touche actuellement détectée.
  » **String | PoolIntArray | Array | PoolStringArray | int id** : Contient le(s) identifiant(s) de(s) touche(s) à ciblée(s). La/les chaîne(s) de caractères fait/font ici référence aux valeurs que doit prendre la clé *key*. Le(s) entier(s) ici fait/font référence aux index de position de la ou des touche(s) pris pour cible dans le champ *Inputs*.

+ **String get_detected_controller () :** Renvoie le nom du contrôleur actuellement détecté. La valeur Unknown Controller est renvoyée lorsqu'un contrôleur n'a pas pu être identifié.

+ **int | PoolIntArray get_keycode (id = null) :** Renvoie le code de la/des touche(s) référencée(s) dans le paramètre *id*. Par défaut, le code de la touche actuellement détectée est renvoyé.
  » **String | PoolIntArray | Array | PoolStringArray | int id** : Contient le(s) identifiant(s) de(s) touche(s) à ciblée(s). La/les chaîne(s) de caractères fait/font ici référence aux valeurs que doit prendre la clé *key*. Le(s) entier(s) ici fait/font référence aux index de position de la ou des touche(s) pris pour cible dans le champ *Inputs*.

+ **float | Vector2 get_normalized_value (id = null) :** Renvoie une valeur d'axe comprise entre 0.0 et 1.0 de la touche référencée dans le paramètre *id*. Par défaut, la valeur d'axe de la touche actuellement détectée est renvoyée. Notez que le changement de type de valeur au niveau de cette fonction se fait par rapport à la valeur contenue dans la clé *type* du champ *Inputs*.
  » **String | int id** : Contient l'identifiant de la touche à ciblée. La chaîne de caractères fait ici référence aux valeurs que doit prendre la clé *key*. L'entier ici fait alusion à l'index de position de la touche pris pour cible dans le champ *Inputs*.

+ **void key_activation (id, activation, delay = 0.0) :** Contrôle l'état d'une touche donnée en terme de permition.
  » **String | PoolIntArray | Array | PoolStringArray | int id** : Contient le(s) identifiant(s) de(s) touche(s) à ciblée(s). La/les chaîne(s) de caractères fait/font ici référence aux valeurs que doit prendre la clé *key*. Le(s) entier(s) ici fait/font référence aux index de position de la ou des touche(s) pris pour cible dans le champ *Inputs*.
  » **bool activation** : Contient l'état de la touche en question.
  » **float delay** : Quel est le temps mort avant le changement d'état ?

+ **bool is_enabled (id) :** Détermine si la ou les touche(s) donnée(s) est/sont active(s) pour d'éventuelle appuies ou relâchements.
  » **String | PoolIntArray | Array | PoolStringArray | int id** : Contient le(s) identifiant(s) de(s) touche(s) à ciblée(s). La/les chaîne(s) de caractères fait/font ici référence aux valeurs que doit prendre la clé *key*. Le(s) entier(s) ici fait/font référence aux index de position de la ou des touche(s) pris pour cible dans le champ *Inputs*.

+ **void enable_keys_without (ids, delay = 0.0) :** Active l'utilisation de toutes les touches prises en charge par le développeur exceptées celles renseignées dans le paramètre *ids*.
  » **Array ids** : Contient les identifiants des touches à ciblées. Chaque élément de ce tableau accepte soit un String (la valeur contenue dans clé *key*), soit un int (l'index de position d'une touche au sein du champ *Inputs*).
  » **float delay** : Quel est le temps mort avant les changements d'états ?

+ **void disable_keys_without (ids, delay = 0.0) :** Désactive l'utilisation de toutes les touches prises en charge par le développeur exceptées celles renseignées dans le paramètre *ids*.
  » **Array ids** : Contient les identifiants des touches à ciblées. Chaque élément de ce tableau accepte soit un String (la valeur contenue dans clé *key*), soit un int (l'index de position d'une touche au sein du champ *Inputs*).
  » **float delay** : Quel est le temps mort avant les changements d'états ?

+ **int | PoolIntArray get_enabled_keys () :** Renvoie les identifants de toutes les touches n'ayant pas été mise hors d'usage.

+ **int | PoolIntArray get_disabled_keys () :** Renvoie les identifiants de toutes les touches ayant été mise hors d'usage.

+ **Dictionary get_enabled_keys_data (json = true) :** Renvoie les données de toutes les touches n'ayant pas été mise hors d'usage.
  » **bool json** : Voulez-vous renvoyer le résultat sous le format json ?

+ **Dictionary get_disabled_keys_data (json = true) :** Renvoie les données de toutes les touches ayant été mise hors d'usage.
  » **bool json** : Voulez-vous renvoyer le résultat sous le format json ?

+ **void set_key_level (id = null, new_value, delay = 0.0) :** Redéfinie la valeur du niveau d'appuie d'une touche donnée. Par défaut, cette méthode cible le niveau d'appuie de la touche actuellement détectée.
  » **String | int id** : Contient l'identifiant de la touche à ciblée. La chaîne de caractères fait ici référence aux valeurs que doit prendre la clé *key*. L'entier ici fait alusion à l'index de position de la touche pris pour cible dans le champ *Inputs*.
  » **int new_value** : Quelle est la nouvelle valeur du niveau d'appuie ?
  » **float delay** : Quel est le temps mort avant la redéfinition ?

+ **int get_key_level (id = null) :** Renvoie la valeur du niveau d'appuie d'une touche donnée. Par défaut, cette méthode renvoie le valeur du niveau d'appuie de la touche actuellement détectée.
  » **String | int id** : Contient l'identifiant de la touche à ciblée. La chaîne de caractères fait ici référence aux valeurs que doit prendre la clé *key*. L'entier ici fait alusion à l'index de position de la touche pris pour cible dans le champ *Inputs*.

+ **bool is_combo (id = null) :** Détermine si une ou plusieurs touche(s) est/sont en traint d'être appuyée(s) de façon répétée. Par défaut, cette méthode renvoie un résultat en se basant sur la touche actuellement détectée.
  » **String | PoolIntArray | Array | PoolStringArray | int id** : Contient le(s) identifiant(s) de(s) touche(s) à ciblée(s). La/les chaîne(s) de caractères fait/font ici référence aux valeurs que doit prendre la clé *key*. Le(s) entier(s) ici fait/font référence aux index de position de la ou des touche(s) pris pour cible dans le champ *Inputs*.

+ **float | Vector2 get_axis_value (id = null) :** Renvoie la valeur d'axe d'une touche grâce à son identifiant. Par défaut, la valeur d'axe de la touche actuellement détectée est renvoyée. Notez que le changement de type de valeur au niveau de cette fonction se fait par rapport à la valeur contenue dans la clé *type* du champ *Inputs*.
  » **String | int id** : Contient l'identifiant de la touche à ciblée. La chaîne de caractères fait ici référence aux valeurs que doit prendre la clé *key*. L'entier ici fait alusion à l'index de position de la touche pris pour cible dans le champ *Inputs*.

+ **int get_key_pourcent (id = null) :** Renvoie en pourcentage la valeur du niveau d'appuie d'une touche grâce à son identifiant. Par défaut, le pourcentage de la valeur d'appuie de la touche actuellement détectée est renvoyée.
  » **String | int id** : Contient l'identifiant de la touche à ciblée. La chaîne de caractères fait ici référence aux valeurs que doit prendre la clé *key*. L'entier ici fait alusion à l'index de position de la touche pris pour cible dans le champ *Inputs*.

+ **bool is_a_button (id = null) :** Détermine si la ou les touche(s) donnée(s) est/sont un/des bouton(s) ou pas. Par défaut, la vérification se fait sur la touche actuellement détectée.
  » **String | PoolIntArray | Array | PoolStringArray | int id** : Contient le(s) identifiant(s) de(s) touche(s) à ciblée(s). La/les chaîne(s) de caractères fait/font ici référence aux valeurs que doit prendre la clé *key*. Le(s) entier(s) ici fait/font référence aux index de position de la ou des touche(s) pris pour cible dans le champ *Inputs*.

+ **bool is_an_axis (id = null) :** Détermine si la ou les touche(s) donnée(s) est/sont un/des bouton(s) d'axe ou pas. Par défaut, la vérification se fait sur la touche actuellement détectée.
  » **String | PoolIntArray | Array | PoolStringArray | int id** : Contient le(s) identifiant(s) de(s) touche(s) à ciblée(s). La/les chaîne(s) de caractères fait/font ici référence aux valeurs que doit prendre la clé *key*. Le(s) entier(s) ici fait/font référence aux index de position de la ou des touche(s) pris pour cible dans le champ *Inputs*.

+ **bool is_pressed (id = null) :** Détermine si la ou les touche(s) donnée(s) est/sont appuyée(s) ou pas. Par défaut, la vérification se fait sur la touche actuellement détectée.
  » **String | PoolIntArray | Array | PoolStringArray | int id** : Contient le(s) identifiant(s) de(s) touche(s) à ciblée(s). La/les chaîne(s) de caractères fait/font ici référence aux valeurs que doit prendre la clé *key*. Le(s) entier(s) ici fait/font référence aux index de position de la ou des touche(s) pris pour cible dans le champ *Inputs*.

IV – Les événements disponibles

+ **key_changed (data) :** Signal déclenché lorsqu'on change de touche dans un procèssus de *Keybindings*. Cet événement renvoie un dictionaire contenant les clés suivantes :
  » **Node node** : Contient le noeud où cet signal a été émit.
  » **String key** : Contient la nouvelle touche détectée.

+ **key_action (data) :** Signal déclenché lorsque la touche détectée respecte les critères exigées par le développeur conduisant ainsi à l'exécution des actions prévues à cet éffet. Cet événement renvoie un dictionaire contenant les clés suivantes :
  » **Node node** : Contient le noeud où cet signal a été émit.
  » **String key** : Contient la touche détectée.
  » **float delta** : Contient le temps éffectué depuis la dernière trame du jeu.

+ **key_cloned (data) :** Signal déclenché lorsque la touche détectée dans un procèssus de *Keybindings* existe déjà dans la liste des touches prise en charge par le module. Cet événement renvoie un dictionaire contenant les clés suivantes :
  » **Node node** : Contient le noeud où cet signal a été émit.
  » **PoolIntArray | int index** : Contient le(s) position(s) de(s) clone(s).
  » **Node | Array other** : Contient le(s) noeud(s) contenant des clones. Cette clé n'est renvoyée que si des répétitions ont été trouvées dans les autres instances de ce module par le biais du champ *ExternalInputs*.

+ **key_level_changed (data) :** Signal déclenché lorsqu'on change le niveau d'appuie d'une touche. Cet événement renvoie un dictionaire contenant les clés suivantes :
  » **Node node** : Contient le noeud où cet signal a été émit.
  » **int value** : Contient la nouvelle valeur du niveau d'appuie.
  » **int index** : Contient l'index de position de la touche qui a été affectée par ce changement.

+ **keydown (data) :** Signal déclenché lorsqu'on appuie sur une touche prise en charge par le module. Cet événement renvoie un dictionaire contenant les clés suivantes :
  » **Node node** : Contient le noeud où cet signal a été émit.
  » **int index** : Contient l'index de position de la touche qui a été appuyée.

+ **keyup (data) :** Signal déclenché lorsqu'on relâche une touche prise en charge par le module. Cet événement renvoie un dictionaire contenant les clés suivantes :
  » **Node node** : Contient le noeud où cet signal a été émit.
  » **int index** : Contient l'index de position de la touche qui a été relâchée.

+ **decreaser_started (data) :** Signal déclenché lorsque le décrémenteur automatique se démarre. Cet événement renvoie un dictionaire contenant la clé suivante :
  » **Node node** : Contient le noeud où cet signal a été émit.

+ **decreaser_stoped (data) :** Signal déclenché lorsque le décrémenteur automatique s'arrète. Cet événement renvoie un dictionaire contenant la clé suivante :
  » **Node node** : Contient le noeud où cet signal a été émit.

+ **key (data) :** Signal déclenché lorsqu'on appuie ou relâche une touche. Cet événement renvoie un dictionaire contenant les clés suivantes :
  » **Node node** : Contient le noeud où cet signal a été émit.
  » **float delta** : Contient le temps éffectué depuis la dernière trame du jeu.
  » **int index** : Contient l'index de position de la touche qui a été appuyée ou relâchée.