# *ControllerSensorFx*

## Table des matières

## I – Définition

**ControllerSensorFx** est un module conçut pour la détection des différentes commandes interagissants avec le jeu. Il adapte les touches d'aide du jeu en fonction du contrôleur détecté. Bien évidemment, ces différentes touches ne sont rien d'autres que des *Sprite*, *Sprite3D* ou *TextureRect* réalisés par le développeur.
Exemple : Le *Sprite* de la touche "X" de la manette XBox ; celui de la touche "Space" du clavier etc...
Le fonctionnement de ce module est assez banal et très simple à exploité.
**NB :** Ce module est de nature indestructible, est compatible à un jeu 2D, 3D et n'est pas sauvegardable.

## II – Les propriétés disponibles

**+ int Controller = 0 :** Contient les différents contrôleurs détectés par le module. La valeur de ce champ varie en fonction des commandes connectées par l'ordinateur.

**+ int Count = -1 :** Combien de commandes voulez-vous écoutée ? La valeur nulle coupe les détections. Tandis que celle négative entraîne une détection ilimitée des commandes.

**+ bool BaseController = true :** Doit-on utilisée la commande de base à chaque déconnexion ?

**+ bool Adapter = false :** Doit-on mettre à jour les *Sprite* des boutons lorsqu'une nouvelle commande est détectée ?

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

» **String category :** Le contrôleur détecté appartient à quelle catégory (xbox 360 ; xbox one ; playstation ; wii ; nintendo switch ; etc...).

» **int search = 3 :** Contient le moyen à utilisé pour chercher les noeuds (*Sprite*, *Sprite3D* ou *TextureRect*) qui seront victime de l'influence de ce module. Les valeurs possibles sont :
-> **MegaAssets.NodeProperty.NAME ou 0 :** Ciblage par nom.
-> **MegaAssets.NodeProperty.GROUP ou 1 :** Ciblage par groupe.
-> **MegaAssets.NodeProperty.TYPE ou 2 :** Ciblage par type.
-> **MegaAssets.NodeProperty.ALL ou 3 :** Ciblage sur n'importe quel type.

» **Vector2 size = Vector2 (50, 50) :** Contrôle la résolution des textures.

» **int quality = 2 :** Contrôle la qualité des textures. Les valeurs possibles sont celles définient au sein de classe *Image* de Godot.

Si vous voulez que la propriété *texture* des noeuds ciblés change en fonction du type de contrôleur détecté par l'ordinateur, vous devez suivre dans le(s) dictionaire(s) défini(en)t au sein de la propriété *Categories*, la nomenclature : *IdentifiantDuNoeud : LienVersImageDuSpriteAssocier*. Si vous désirez affecter une même texture à plusieurs noeuds à la fois, séparez chaque identifiant de noeud par un pipe. IDEM pour la catégory.

**+ PoolStringArray TargetScenes :** Cette option permet au module de savoirs quant est-ce qu'il doit ciblé des noeuds ou pas. N'ajoutez que les scènes possédant des noeuds ayant pour objectif de servir de bouton d'aide à l'utilisateur de votre produit. Cela permettra ainsi d'optimiser les recherches à faire avant la mise à jour de la texture des noeuds pris pour cible.
**NB :** La présence de doublons au niveau des catégories de manettes et les noms des scènes n'est pas tolérée.

## III – Les méthodes disponibles

**+ PoolStringArray get_detected_controller_names () :** Renvoie les noms de tous les contrôleurs se trouvant dans l'intervalle de détection imposé par le développeur. Pour agir sur l'intervalle de détection, modifiez la valeur de la propriété *Count*.

**+ String get_active_controller_name () :** Renvoie le nom du contrôleur en cours d'utilisation.

**+ int get_active_controller_index () :** Renvoie la position du joueur associé au contrôleur en cours d'utilisation.

## IV – Les événements disponibles

**+ controller_changed (name) :** Signal déclenché lorsqu'on change de contrôleur.
» **String name :** Contient le nom du nouveau contrôleur actif.

**+ controller_connected (name) :** Signal déclenché lorsqu'un nouveau contrôleur a été connecté à l'ordinateur.
» **String name :** Contient le nom du nouveau contrôleur connecté.

**+ controller_disconnected (name) :** Signal déclenché lorsqu'un nouveau contrôleur a été déconnecté de l'ordinateur.
» **String name :** Contient le nom du contrôleur déconnecté.