InputGeneratorFx

Table des matières

I – Définition

InputGeneratorFx est un module conçut pour les lectures de commandes programmées. Son importance prend vie lorsqu'on souhaite établir une certaine interaction entre le joueur et un *IA*. Ces interactions seront souvent matérialisées par des animations entre le joueur et le système du jeu. Ce module génère de façon aléatoire une touche en fonction du type de contrôleur détecté dont l'utilisateur devra appuyer avant un délai donné. Lorsque la touche générée est appuyée, une action se déclenche caractérisant la réussite. Dans le cas contraire, une autre action se déclenche caractérisant l'échec. Ainsi, l'utilisateur peut décidé de ce qui se passera en cas de réussite et d'échec. Notez que se module prend également en charge les appuies répétés sur une même touche (*Combo*).
NB : Ce module est compatible à un jeu 2D, 3D et n'est pas sauvegardable. Avant de continuer, le module ne supporte pas encore les écrans tactiles.

II – Les propriétés disponibles

+ int PlayerIndex = 0 : Quel joueur voulez-vous écouter ? Notez que cette option n'est destiné qu'aux manettes.

+ 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 :
-> **InputGeneratorFx.KEYBOARD ou 0** : Ecoute le clavier de l'ordinateur.
-> **InputGeneratorFx.MOUSE ou 1** : Ecoute la souris de l'ordinateur.
-> **InputGeneratorFx.JOYSTICK ou 2** : Ecoute la manette connectée à l'ordinateur.
-> **InputGeneratorFx.INPUTMAP ou 3** : Ecoute la carte des entrées définit au sein de Godot.

» **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 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. Notez que cela se fera chaque fois qu'une touche sera générée et cette option n'est pas activée lorsque le contrôleur utilise la carte des entrées définit au sein de Godot.

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

» **float delay = 0.0** : Quel est le temps mort avant l'exécution du défis en question ? Cette option ne s'active que lorsque sa valeur est supérieur à **0.0**.

» **float time = 0.0** : Combien de temps, le défis durera t-il ? Cette option ne s'active que lorsque sa valeur est supérieur à **0.0**. Dans le cas contraire, le défi attend sa validation.

» **int repeat = 1** : Combien de fois, le défis va t-il se répété ? Cette option ne s'active que lorsque sa valeur est supérieur à **0**. Dans le cas contraire, le défi n'est pas exécuté.

» **int slow = 0** : Contrôle le comportement des éffets de ralentissements dans un défi. Les valeurs possibles sont :
-> **InputGeneratorFx.SlowMotion.NONE ou 0** : Aucun effet de ralentissement ne se déclenchera.
-> **InputGeneratorFx.SlowMotion.PIG_PONG ou 1** : L'effet de ralentissement se déclenchera de manière alterné.
-> **InputGeneratorFx.SlowMotion.RANDOM ou 2** : L'effet de ralentissement se déclenchera de façon aléatoire.
-> **InputGeneratorFx.SlowMotion.ALL ou 3** : L'effet de ralentissement se déclenchera lorsque le défi sera lancer.

» **float | Vector2 rate = 0.3** : Quel sera le taux ou le degré de l'éffet de ralentissement à son déclenchement ? Si un réel est utilisé, l'éffet se déclenchera avec le taux donné par le développeur. Si un vecteur deux dimention est utilisé, l'éffet se déclenchera avec un taux appartenant à l'intervalle précisé soit [0.0 ; 1.0]. Dans le cas contraire, aucun taux ne sera appliqué.

» **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 | Dictionary success** : Que se passera t-il lorsque le défi imposé a été réussit par son joueur ? 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 | Dictionary failed** : Que se passera t-il lorsque le défi imposé a été échoué ? Notez que cette dernière possède les mêmes propriétés que la clé *success* et s'utilise de la même manière que lui.

NB : Les répétitions au niveau des touches ne sont pas tolérées.

III – Les méthodes disponibles

+ **void start_challenges (delay = 0.0) :** Démarre l'exécution des défi prises en charge par le module.
  » **float delay** : Quel est le temps mort avant le démarrage des défi ?

+ **void stop_challenges (delay = 0.0) :** Annule l'exécution des défi prises en charge par le module. Cette fonction n'agit que lorsque les défi imposés sont déjà en cours d'exécution.
  » **float delay** : Quel est le temps mort avant l'annulation des défi ?

+ **void pause_challenges (delay = 0.0) :** Suspend l'exécution des défi prises en charge par le module. Cette fonction n'agit que lorsque les défi imposés sont déjà en cours d'exécution.
  » **float delay** : Quel est le temps mort avant la suspension des défi ?

+ **bool is_combo () :** Détermine si la touche actuellement détectée est en traint d'être appuyée de façon répétée ou pas.

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

+ **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 get_generated_key () :** Renvoie l'index de position de la touche qui a été générée avant l'exécution d'un défi.

+ **int get_preview_key () :** Renvoie l'index de position de la touche qui a été utilisée précédement.

+ **bool is_running () :** Les défi sont-ils en cours d'exécution ?

+ **bool get_challenge_index () :** Renvoie l'index de position du défi actuellement en cours d'exécution.

+ **float get_normalized_value () :** Renvoie une valeur d'axe comprise entre 0.0 et 1.0 de la touche actuellement détectée.

+ **int get_keycode () :** Renvoie le code de la touche actuellement détectée.

+ **int get_key_level () :** Renvoie la valeur du niveau d'appuie de la touche actuellement détectée.

+ **int get_key_pourcent () :** Renvoie en pourcentage la valeur du niveau d'appuie de la touche actuellement détectée.

+ **bool is_a_button () :** Détermine si la touche actuellement détectée est un bouton ou pas.

+ **bool is_an_axis () :** Détermine si la touche actuellement détectée est un bouton d'axe ou pas.

+ **int get_current_repeat () :** Renvoie le nombre actuel de fois que le défi actuel à été exécuté.

IV – Les événements disponibles

+ **challenge_started (data) :** Signal déclenché juste au démarrage d'un défi. Cet événement renvoie un dictionaire contenant les clés suivantes :
  » **Node node** : Contient le noeud où cet signal a été émit.
  » **int count** : Contient le nombre de fois que le défi en question a été répété.
  » **int challenge** : Contient l'index de position du défi actuellement en cours d'exécution.

+ **challenge_finished (data) :** Signal déclenché à la fin d'un défi. Cet événement renvoie un dictionaire contenant les clés suivantes :
  » **Node node** : Contient le noeud où cet signal a été émit.
  » **int count** : Contient le nombre de fois que le défi en question a été répété.
  » **int challenge** : Contient l'index de position du défi actuellement en cours d'exécution.

+ **challenge_changed (data) :** Signal déclenché lorsque le défi actuel a changé. Cet événement renvoie un dictionaire contenant les clés suivantes :
  » **Node node** : Contient le noeud où cet signal a été émit.
  » **int count** : Contient le nombre de fois que le défi en question a été répété.
  » **int challenge** : Contient l'index de position du défi actuellement en cours d'exécution.

+ **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 détectée.

+ **keydown (data) :** Signal déclenché lorsqu'on appuie sur une touche. 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. 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 les clés suivantes :
  » **Node node** : Contient le noeud où cet signal a été émit.
  » **int challenge** : Contient l'index de position du défi actuellement en cours d'exécution.

+ **decreaser_stoped (data) :** Signal déclenché lorsque le décrémenteur automatique s'arrête. Cet événement renvoie un dictionaire contenant les clés suivantes :
  » **Node node** : Contient le noeud où cet signal a été émit.
  » **int challenge** : Contient l'index de position du défi actuellement en cours d'exécution.

+ **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.
  » **int index** : Contient l'index de position de la touche qui a été appuyée ou relâchée.

+ **success (data) :** Signal déclenché lorsque le joueur valide les conditions exigées par un défi. Cet événement renvoie un dictionaire contenant les clés suivantes :
  » **Node node** : Contient le noeud où cet signal a été émit.
  » **int count** : Contient le nombre de fois que le défi en question a été répété.
  » **int challenge** : Contient l'index de position du défi actuellement en cours d'exécution.

+ **failed (data) :** Signal déclenché lorsque le joueur n'a pas pu validé les conditions exigées par un défi. Cet événement renvoie un dictionaire contenant les clés suivantes :
  » **Node node** : Contient le noeud où cet signal a été émit.
  » **int count** : Contient le nombre de fois que le défi en question a été répété.
  » **int challenge** : Contient l'index de position du défi actuellement en cours d'exécution.