Table des matières

I – Définition

InverseKinematicFx est un module conçut pour le pistage d'un objet à partir d'un squellette d'animation. Ce module oblige la section du squellette sélectionnée pour cibler un objet.
NB : Ce module est compatible à un jeu 3D et n'est pas sauvegardable. Il possède également un pisteur permettant de pister automatiquement ces cibles en fonction de son champ de vision comme pour le module *CameraControlFx*.

II – Les propriétés disponibles

+ **NodePath TargetSkeleton :** Contient l'instance d'un noeud de type *Skeleton*.

+ **String BoneName :** Quelle section du squellette d'animation voulez-vous ciblée ?

+ **int LookAtAxis = 2 :** Sur quelle direction les pistages se feront ? Les valeurs possibles sont :
-> **MegaAssets.Axis.X ou 1 :** L'axe des absisses.
-> **MegaAssets.Axis.Y ou 2 :** L'axe des ordonnés.
-> **MegaAssets.Axis.Z ou 3 :** L'axe des côtes.
-> **MegaAssets.Axis._X ou 4 :** L'opposé de l'axe des absisses.
-> **MegaAssets.Axis._Y ou 5 :** L'opposé de l'axe des ordonnés.
-> **MegaAssets.Axis._Z ou 6 :** L'opposé de l'axe des côtes.

+ **int FroozeAxis = 0 :** Quel axe bloqué au cours des pistages ? Les valeurs possibles sont :
-> **MegaAssets.Axis.NONE ou 0 :** Aucun blockage.
-> **MegaAssets.Axis.X ou 1 :** Blockage de l'axe des absisses.
-> **MegaAssets.Axis.Y ou 2 :** Blockage de l'axe des ordonnés.
-> **MegaAssets.Axis.Z ou 3 :** Bockage de l'axe des côtes.
-> **MegaAssets.Axis._X ou 4 :** Blockage de l'opposé de l'axe des absisses.
-> **MegaAssets.Axis._Y ou 5 :** Blockage de l'opposé de l'axe des ordonnés.
-> **MegaAssets.Axis._Z ou 6 :** Blockage de l'opposé de l'axe des côtes.
-> **MegaAssets.Axis.XY ou 7 :** Bockage des axes x et y.
-> **MegaAssets.Axis.XZ ou 8 :** Bockage des axes x et z.
-> **MegaAssets.Axis.YZ ou 9 :** Bockage des axes y et z.
-> **MegaAssets.Axis._XY ou 10 :** Bockage de l'opposé des axes x et y.
-> **MegaAssets.Axis._XZ ou 11 :** Bockage de l'opposé des axes x et z.
-> **MegaAssets.Axis._YZ ou 12 :** Bockage de l'opposé des axes y et z.
-> **MegaAssets.Axis.XYZ ou 13 :** Bockage des axes x, y et z.
-> **MegaAssets.Axis._XYZ ou 14 :** Bockage de l'opposé des axes x, y et z.

+ **Vector3 Offset = Vector3 (0.0, 0.0, 0.0) :** Contrôle l'ajustement de la section du squellette d'animation en terme de rotation.

+ **NodePath ListenCamera :** Contient l'instance d'un noeud de type *CameraControlFx*. Si l'on statisfait l'entrée qu'attend cette propriété, le module se synchrônisera aux données renvoyées par le module *CameraControlFx* uniquement lorsque ce dernier activera son système de pistage automatique.

+ **NodePath TargetingAera :** Contient l'instance d'un noeud de type *Area* représentant le champ d'activité du module. Si l'entrée de cette propriété n'est pas satisfaite, le module cherchera toutes les références des objets ou des noeuds donnés par le développeur à partir des configurations éffectuées dans le champ *Targets*.

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

» **String id** : Quel est l'identifiant du noeud à prendre en charge ? L'utilisation de cette clé est obligatoire.

» **int search = 3** : Quel moyen utilisé pour chercher le noeud à prendre en charge ? Notez que l'identifiant donné est pisté à par un programme de recherche. Les valeurs possibles sont :
-> **MegaAssets.NodeProperty.NAME ou 0** : Trouve un noeud en utilisant son nom.
-> **MegaAssets.NodeProperty.GROUP ou 1** : Trouve un noeud en utilisant le nom de son groupe.
-> **MegaAssets.NodeProerty.TYPE ou 2** : Trouve un noeud en utilisant le nom de sa classe.
-> **MegaAssets.NodeProerty.ANY ou 3** : Trouve un noeud en utilisant l'un des trois moyens cités plus haut.

» **bool ignored = false** : Le pisteur de la caméra doit-il ignoré l'identifiant précisé ?

» **float transition = 1.0** : Combien de temps prend le passage d'une cible à une autre ?

» **int type = 0** : Quel type de transition adopté ? Les valeurs possibles de ce champ sont celles de Godot. Cette propriété est solicité au changement de cible.

» **int easing = 2** : Quel assouplissement adopté ? Les valeurs possibles de ce champ sont celles de Godot. Cette propriété est solicité au changement de cible.

» **Array | Dictionary entered** : Signal déclenché lorsque l'objet entre dans le champ de vision du pisteur. Cette clé exécute les différentes actions données à son déclenchement. Pour soumettre les actions à exécutées référez vous à la méthode utilisée au niveau de la clé *actions* de la propriété *EventsBindings* dans les bases du framework.

» **Array | Dictionary exited** : Signal déclenché lorsqu'un objet sort du champ de vision du pisteur. Cette clé exécute les différentes actions données à son déclenchement. Pour soumettre les actions à exécutées référez vous à la méthode utilisée au niveau de la clé *actions* de la propriété *EventsBindings* dans les bases du framework.

III – Les méthodes disponibles

+ **void change_target (index = -1, delay = 0.0) :** Force la section du squellette d'animation à changé de cible parmit celles détectées. Par défaut, une cible est générée si l'index de la nouvelle cible n'a pas été donné.
  » **int index** : Contient l'index de la nouvelle cible à pistée.
  » **float delay** : Quel est le temps mort avant le changement ?

+ **Dictionary get_targets_data (json = false) :** Renvoie toutes les données concernant les cibles de l'inverseur kinénatique.
  » **bool json** : Voulez-vous renvoyer les données au format json ?

+ **Node get_current_target () :** Renvoie la référence de l'objet ou du noeud actuellement ciblé par le sytème de pistage automatique.

+ **Node get_preview_target () :** Renvoie la référence de l'objet ou du noeud ayant été précédement ciblé par le sytème de pistage automatique.

+ **Node get_next_target () :** Renvoie la référence du future objet ou noeud qui sera ciblé par le sytème de pistage automatique.

IV – Les événements disponibles

+ **target_changed (data) :** Signal déclenché lorsque le pisteur a changé de cible. Cet événement renvoie un dictionaire contenant les clés suivantes :
  » **Node node** : Contient le noeud où ce signal a été émit.
  » **Node target** : Contient la référence de la nouvelle cible du pisteur.

+ **target_entered (data) :** Signal déclenché lorsqu'un objet entre dans le champ de vision du pisteur. Cet événement renvoie un dictionaire contenant les clés suivantes :
  » **Node node** : Contient le noeud où ce signal a été émit.
  » **Node target** : Contient la référence de l'objet détecté.

+ **target_exited (data) :** Signal déclenché lorsqu'un objet sort du champ de vision du pisteur. Cet événement renvoie un dictionaire contenant les clés suivantes :
  » **Node node** : Contient le noeud où ce signal a été émit.
  » **Node target** : Contient la référence de l'objet détecté.

+ **target_generated (data) :** Signal déclenché lorsqu'une cible a été générée par le pisteur. Cet événement renvoie un dictionaire contenant les clés suivantes :
  » **Node node** : Contient le noeud où ce signal a été émit.
  » **Node target** : Contient la référence de la future cible du pisteur.