Table des matières

I – Définition

PermutatorFx est un module conçut pour permuter un noeud par un autre. La permutation peut se faire en fonction de certains critères.
NB : Ce module est compatible à un jeu 2D, 3D et est sauvegardable.

II – Les propriétés disponibles

+ **NodePath Target :** Contient l'instance d'un noeud de type *Spatial* ou *Node2D*.

+ **NodePath Domain :** Contient l'instance d'un noeud de type *Area* ou *Area2D*. Notez que si la cible est un objet appartenent à un plan, alors vous devriez choisir un domaine du plan. IDEM pour l'objet dans l'espace.

+ **bool PartialDestruction = false :** Doit-on détruire la cible par la méthode *queue_free ()* ou par la propriété *visible*.

+ **float Solidity = 0.0 :** Contient la solidité à appliquée à la cible avant sa permutation. La valeur de cette propriété est dans l'intervalle [0.0, 100.0]. C'est aussi son degré de résistance.

+ **Array Subtitutes :** Tableau de dictionnaires ou de chaînes contenant toutes les différentes configurations sur chaque descendant prise en charge par le développeur. Cette propriété renferme tous les objets qui devront remplacer la cible de ce module. Les dictionnaires issus de ce tableau supportent les clés suivantes :

» **String path :** Quel est le chemin pointant vers le remplaçant de la dite cible ? L'utilisation de cette clé est obligatoire.

» **Vector3 | Vector2 | NodePath position :** A quelle position sera importer le remplaçant de la cible du module ? Si vous donnez le chemin d'un noeud de la scène, veillez à ce que l'objet soit (dans le plan si la cible donnée dans le champ *Target* est un noeud de type *Node2D*, dans l'espace sinon). Notez que la non utilisation de cette clé provoquera une importation qui ciblera la position qu'avait son pré-décésseur avant sa destruction.

» **bool rotated = true :** Doit-on changer la rotation du remplaçant à celle de son pré-décésseur ?

» **bool background = true :** Contrôle le moyen utilisé pour charger un objet. A *true*, le chargement de l'objet s'éffectue en arrière plan sans bloqué le jeu.

» **bool import = true :** Doit-on importer automatiquement le remplaçant ? Cette propriété n'est solicité qu'après le chargement des données du gestionnaire des données du jeu.

» **bool physic = false :** Doit-on projeté le remplaçant physiquement en fonction de puissance et de

la direction de frappe ?

+ **Array Triggers :** Tableau de dictionnaires contenant toutes les différentes configurations sur chaque objet prise en charge par le développeur. Cette propriété renferme tous les objets qui devront percuter la cible de ce module. 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** : Doit-on ignoré l'identifiant précisé ? Dans ce cas, l'objet portant l'identifiant spécifié sera ignoré à sa détection.

» **float power = 0.0** : Contient la puissance de frappe de l'objet qui percutera la cible qu'écoute le module. Notez que plus la force de percution est forte plus vite la cible sera remplacée par ces descendants.

III – Les méthodes disponibles

+ **void is_permute () :** La cible actuelle a t-elle été déjà permutée ?

IV – Les événements disponibles

+ **before_permute (node) :** Signal déclenché avant la permutation d'un objet par un autre.
» **Node node** : Contient le noeud où ce signal a été émit.

+ **after_permute (node) :** Signal déclenché après la permutation d'un objet par un autre.
» **Node node** : Contient le noeud où ce signal a été émit.

+ **collision (data) :** Signal déclenché lorsqu'un objet prise en charge par le module percute sa cible. Notez que cela ne s'applique qu'aux objets non ignorés par le module. Cet événement renvoie un dictionaire contenant les clés suivantes :
» **Node node** : Contient le noeud où ce signal a été émit.
» **Variant object** : Contient la référence de l'objet qui a été détecté. L'objet renvoyé sera l'instance d'un noeud de type *Spatial* ou *Node2D*.