# SkillTree documentation v1.5

## *Vue d'ensemble*

Le système de SkillTree se divise en 6 widgets: le WidgetSkillNode, le WidgetLine le WidgetSkillDescriptor, le WidgetSkillTreeCanvas, le WidgetSkillTreeCanvasContainer et le WidgetSkillTree

Le WidgetSkillNode correspond à un noeud dans le skilltree. Il est relié à un skill, et possède ces conditions de déblocage.

Le WidgetLine assure l'affichage d'un lien entre deux WidgetSkillNode

Le WidgetSkillDescriptor permet d'afficher les informations relatives à un skill

Le WidgetSkillTreeCanvas est le widget qui permet de construire le skilltree

Le  WidgetSkillTreeCanvasContainer est le widget qui permet d'afficher un WidgetSkillTreeCanvas, et est responsable du zoom et du positionnement de ce dernier.

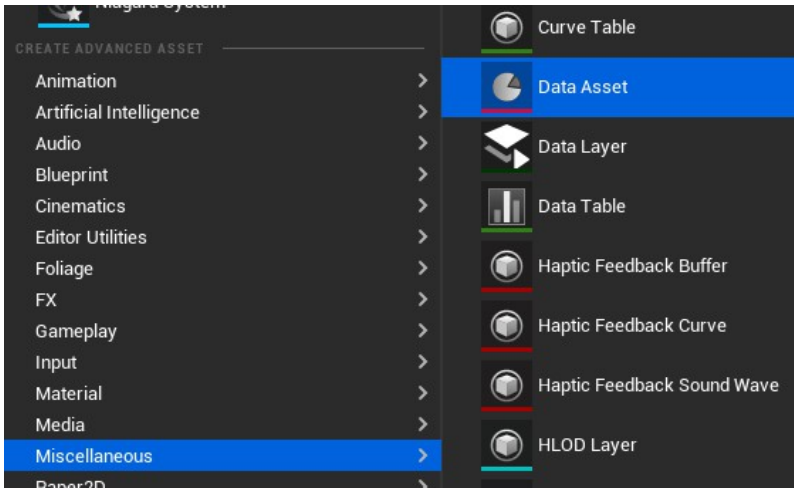
Le WidgetSkillTree est le widget qui est afficher en jeu, reliant les fonctionnalités des différents widgets entre eux.

Enfin, un skill est un data asset qui en fonction de la statistique qui doit changer contiendra plusieurs variables.

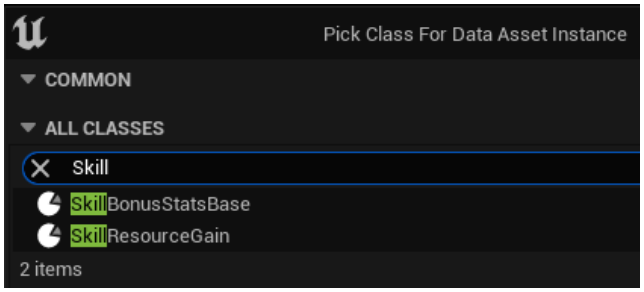
# *Construire un Skill*

## Créer un Skill

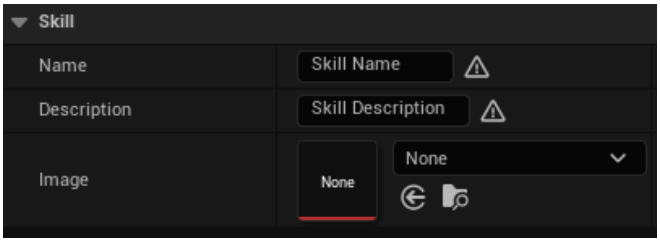
Un Skill est un simple data asset contenant différentes variables qui peuvent changer plusieurs aspect du jeu. Plusieurs types de skills sont disponibles, chacun avec différents effets.




Pour en créer un, il suffit de faire clic droit dans le content drawer, aller dans Miscellaneous, puis de sélectionner Data Asset.




Dans la fenêtre qui viens de s'ouvrir, il suffit maintenant de taper « Skill » dans la barre de recherche, puis de sélectionner le type de Skill que vous voulez créer.




Peu importe le type de skill créer, ils ont tous en commun trois variables :

- Name : Le nom du Skill
- Description : Une description de l'effet du Skill
- Image : Une image pour représenter le Skill sur le SkillTree

N'oublier pas d'activer la localisation si le nom ou la description doivent être traduits !

## Les différents types de Skills

### *Skill Resource Gain*

Ce skill permet de donner immédiatement au joueur une certaine quantité de ressources.

Stone, Wood et Food Gain donne respectivement la quantité indiqué de pierre, bois et nourriture.

### *Skill Bonus Stats*

Ce skill permet de modifier les stats basiques à une ou plusieurs entité.

La variable Concerned Entities permet de choisir quelles entités doivent être affecté par le changement de stats et avec quels bonus. (Pour créer des stats bonus, voir la prochaine rubrique)

### *Skill Army Count Stat Bonus*

Ce skill permet d'appliqué un bonus à une ou des entités si le joueur a un certain nombre d'unités armées.

La variable Concerned Entities permet de choisir quelles entités doivent être afecté par le changement de stats et avec quels bonus. (Pour créer des stats bonus, voir la prochaine rubrique)

La variable Army Count Trigger représente le nombre d'unités armés que le joueur doit posséder pour que les bonus soient appliqué. S'il en possède plus, le bonus sera appliqué, et s'il est en-dessous le bonus sera enlevé.

### *Skill Max Pop Raise*

Ce skill permet d'augmenter la population maximale que peut atteindre le joueur.

La variable Max Pop Raise représente la population maximale à ajouter.

### *Skill Zone Point Gain*

Ce skill permet de donner immédiatement au joueur une certaine quantité de zone points.

Zone Points défini le nombre de points à accorder.

### *Skill Unlock Building*

Ce skill permet de debloquer des bâtiments que le joueur pourra construire.

La variable Building To Unlock permet de choisir quel est le bâtiment à débloquer.

### *Skill Unlock Entity*

Ce skill permet de debloquer des unités que le joueur pourra former.

La variable Entity To Unlock permet de choisir quel est l'unité à débloquer.

## Créer des stats bonus

Les stats bonus s'appliquent avec des blueprints de Stats, exactement les même qui sont utilisé pour définir les stats par défaut de toute entité. Cépendant, toutes les unités ne possèdent pas les même stats, et il peut être donc difficile de retrouver qui possède quoi. Afin de faciliter la recherche de quel type se stats à créer, voici les différents types de stats disponibles. A noter que certain Stats héritent d'autre Stats. Cela veux dire quelle possède toutes les variables du Stat dont elle hérite.

### *Stats*

Les stats de bases que possèdent toutes entités.

Les variables Name, Description et Image ne sont pas améliorable et peuvent être ignorées.

Max life: Améliore la vie maximale de l'entité (en %)

Range view: Améliore la porté de vision de l'entité (en %)

### *Skeletal Stats*

Hérite de Stats

Les stats que possèdent les unités

La variable Refresh Rate n'est pas améliorable et peut donc être ignorée.

Speed: Améliore la vitesse de l'entité (en %)

### *Priced Skeletal Stats*

Hérite de Skeletal Stats

Les stats que possèdent les unités possédant un prix

Wood Price: Baisse le prix en bois de l'unité (en %)

Food Price: Baisse le prix en nourriture de l'unité (en %)

Stone Price: Baisse le prix en pierre de l'unité (en %)

Population Price: Baisse le prix en population de l'unité (en %)

### *Builder Stats*

Hérite de Priced Skeletal Stats

Les stats que possèdent les unités constructeurs

Ne possède aucune stats pour l'instant

*Worker Stats*

Hérite de Priced Skeletal Stats

Les stats que possèdent les unités recuillant des ressources

Harvest Time: Réduit le temps de récolte (en %)

Resource Capacity: Augmente la capacité totale que l'unité peut porte (en brut)

*Army Stats*

Hérite de Priced Skeletal Stats

Les stats que possèdent les unités armé

Les variables Attack When Hit Percentage, Attack In Range Percentage et Wait Time ne sont pas améliorable et peuvent être ignorées

Formation Time: Réduit le temps de formation (en %)

Attack Speed: Augmente la vitesse d'attaque (en %)

Attack Range: Augmente la portée d'attque (en %)

Attack Strength: Augmente les dégâts infligés (en %)

Defense Range: Augmente la portée de défense (en %)

*Heal Army Stats*

Hérite de Army Stats

Les stats que possède une unité armé capable de soigner

Heal Strength: Augmente la puissance de soin (en %)

Heal Speed: Augmente la vitesse de soin (en %)

*Ranged Army Stats*

Hérite de Army Stats

Les stats que possède une unité armée capable d'attaquer avec des projectiles

Les variables Projectile, Projectile Speed et Projectile Max Height ne sont pas améliorables et peuvent donc être ignorées

Has Area Of Effect: Le projectile tiré doit-il gagner une zone d'effet (Vrai/Faux)

Area Of Effect Range: Augmente la taille de la zone d'effet (en %)

*Building Stats*

Hérite de Stats

Les stats que possède un bâtiment

Les variables Width et Length ne sont pas améliorable et peuvent être ignorées

*Priced Building Stats*

Hérite de Building Stats

Les stats que possède un bâtiment possédant un prix

La variable Life Regain n'est pas améliorable et peut donc être ignorée

Wood Price: Baisse le prix en bois du bâtiment (en %)

Food Price: Baisse le prix en nourriture du bâtiment (en %)

Stone Price: Baisse le prix en pierre du bâtiment (en %)

Population Price: Baisse le prix en population du bâtiment (en %)

Wood Repair Price: Baisse le prix en bois de la réparation du bâtiment (en %)

Food Repair Price: Baisse le prix en nourriture de la réparation du bâtiment (en %)

Stone Repair Price: Baisse le prix en pierre de la réparation du bâtiment (en %)

Repair Time: Réduit le temps nécessaire à la réparation du bâtiment (en %)

Build Time: Réduit le temps nécesaire à la construction du bâtiment (en %)

Workforce Capacity: Augmente la capacité maximale d'entité du bâtiment (en brut)

*Defence Building Stats*

Hérite de Priced Building Stats

Les stats que possèdent un bâtiment défensif

Les variables Projectile, Projectile Max Height et Max Unit Capacity ne sont pas améliorable et peuvent donc être ignorées

Range: Augmente la porté d'attaque du bâtiment (en %)

Fire Rate: Augmente la rapidité d'attaque du bâtiment (en %)

Has Area Of Effect: Le projectile tiré doit-il gagner une zone d'effet (Vrai/Faux)

Area Of Effect Range: Augmente la taille de la zone d'effet (en %)

*Harvest Building Stats*

Hérite de Priced Building Stats

Les stats que possèdent un bâtiment de production de ressources

- Generated Resource: Augmente la quantité de resources généré (en %)
- Generated Resource Timer: Réduit le temps de génération de ressources (en %)
- Is Upgradable: Le bâtiment peut-il être améliorer (Vrai/Faux)

*Villager Building Stats*

Hérite de Priced Building Stats

Les stats que possèdent un bâtiment civil

- Villager Create: Augmente le nombre de villageois créer (en brut)
- Builder Capacity: Augmente le nombre de constructeurs venant de ce bâtiment (en brut)

*Heal Building Stats*

Hérite de Priced Building Stats

Les stats que possèdent un bâtiment qui à la capacité de soigner

- Heal Strength: Augmente la puissance de soin (en %)
- Heal Speed: Augmente la vitesse de soin (en %)

*Enemy Building Stats*

Hérite de Building Stats

Les stats que possèdent un bâtiment enemie

- Ne possède aucune stats pour l'instant

*Resource Stats*

Hérite de Stats

Les stats que possèdent une ressource

- Ne possède aucune stats pour l'instant

# *Constuire un SkillTree*

## SkillNode

Avant de commencer le SkillTree, il faut pouvoir représenter un skill seul, ce qui est le rôle du SkillNode.

Il nécessite 3 widgets obligatoires:

- Une image “skillImage” qui affichera la texture du skill assigné
- Un button “button” qui détecte le clique du joueur
- Un text “skillNumText” qui affiche le nombre de skills à acquérir avant de pouvoir débloquer ce skill.

Il possède aussi certaine variables

- Skill: Une réference au skill que le noeud répresente
- Previous Skills: Un tableau contenant les précédants skills qui peuvent le débloquer ainsi que le type de WidgetLine à utiliser pour les relier
- Num Skills Required: Le nombre de skill que le joueur doit posséder avant de pouvoir le débloquer
- Threshold: Le palier auquel ce skill appartient. Lorsqu'il est activé, il activera le skill dans SkillTree, à part si un autre SkillNode avec le même threshold à déjà été activé.
- Locked/Unlocked/Acquired Color: Les couleurs que penderont le skill s'il est bloqué/débloqué/acquéri

Penser bien que ces variables seront les valeurs par défaut quand un SkillNode est créer dans le SkillTree. Il est donc préférable de changer uniquement les différentes couleurs. Les autres variables seront définies dans la prochaine étape.

## WidgetLine

Le WidgetLine est l'élément qui en jeu affichera le lien entre deux SkillNode.

Il possède 3 variables:

- Thickness: La largeur de la ligne
- Active Color: La couleur de la ligne lorsque le précédent SkillNode est acquéri
- Inactive Color: La couleur de la ligne loreque le précédent SkillNode n'est pas acquéri

De plus, il y a plusieurs versions de WidgetLine en fonction du type de ligne voulu.

### *WidgetStraightLine*

Ce widget créer une simple ligne droite

### *WidgetBifurcatedLine*

Ce widget créer une ligne qui sera toujours vertical ou horizontal, biffurquant à un point donné. Ce point est calculé avec la variable “Bifurcation Point”, une valeur normalisée entre 0 et 1 correspondant à un point entre le départ et l'arrivée. Il est aussi possible de choisir s'il doit commencer verticalement ou horizontalement avec la variable “Is Horizontal”

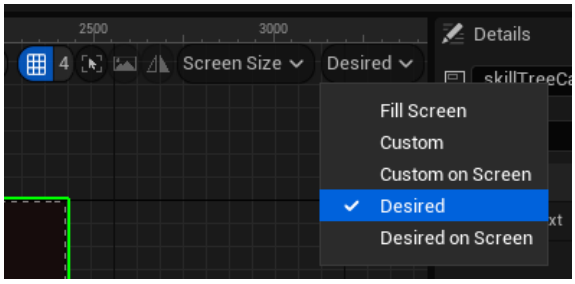
### *WidgetInvisibleLine*

Ce widget ne créra pas de lien visible en jeu. Il est surtout utile pour relier deux SkillNodes dont le lien est implicite.

## SkillTreeCanvas

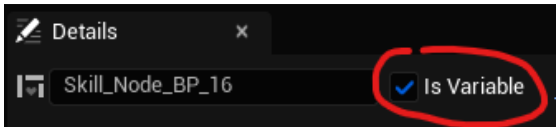
Le SkillTreeCanvas est le widget où le SkillTree est construit. C'est ici que les SkillNodes sont utilisés et reliés.

Il nécessite qu'un seul widget obligatoire, un Canvas Panel "skillTreeCanvas" qui contiendra tous les SkillNodes.



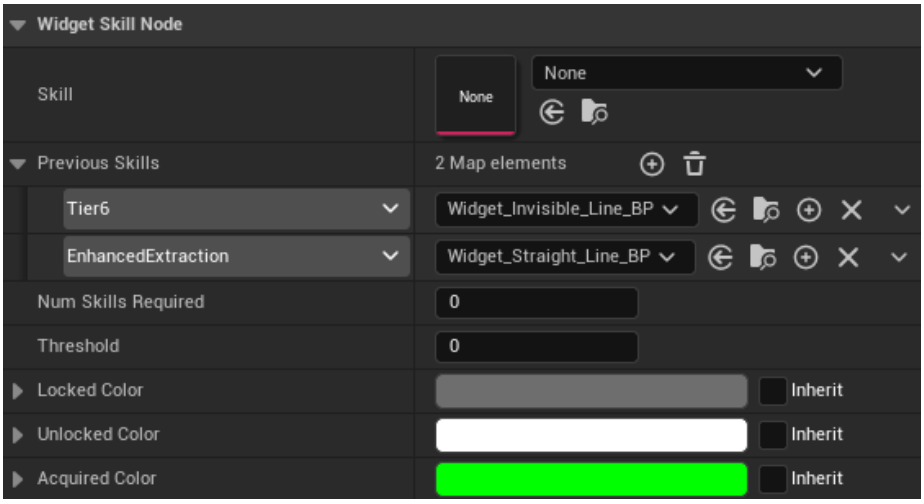

Ce Canvas Panel est fait pour d'adapter à son contenu. Pour voir cette adaptation en éditeur, il suffit de choisir « Desired » dans le paramètre en haut à droite.

Ensuite, il suffit de mettre autant de SkillNodes que vous voulez où vous voulez. Une fois placé dans le canvas, il faut penser à définir le SkillNode comme étant une variable, en haut à droite de l'éditeur, sinon il ne pourras pas être relié à d'autres SkillNodes.




Il est aussi hautement recommander de ne pas laisser le nom par défaut et de renommer les SkillNodes de façon reconnaissable. Ceci facilitera la liaison entre les différent SkillNodes.

Ensuite, il faut pouvoir assigner un skill au SkillNode, mais aussi définir ses conditions de déblocages. Ceci se fait dans leurs variables.



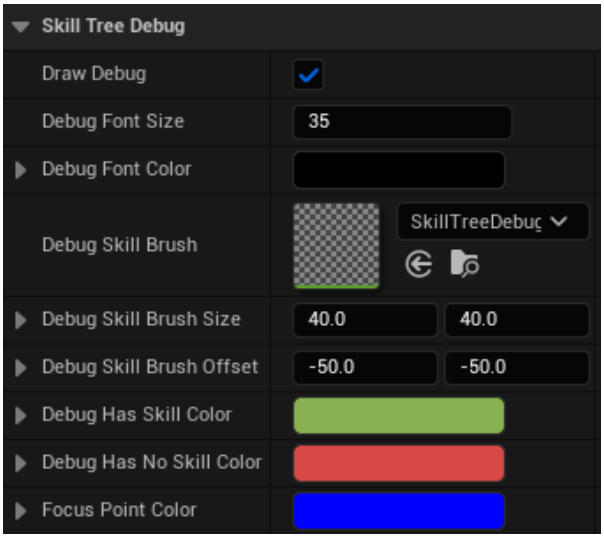

Locked, Unlocked et Acquired Color peuvent être défini dans le Blueprint du SkillNode, mais si besoin vous pouvez les changer ici aussi. Ce qui importe le plus ici sont les variables Skill, Previous Skills et Num Skills Required.

Skill est une référence a un skill créer au préalable, non pas un SkillNode.

Previous Skills représente les SkillNodes derrières lesquels ce SkillNode est bloqué. Pour rajouter un skill, il suffit de cliquer sur le boutton plus. Le nouvel élément créer demandera deux arguments. Le premier est le SkillNode avec qui il doit être relié. Pensez bien que plus il y a de SkillNode dans le canvas, plus la liste sera longue, d'où l'utilité de les nommer proprement. Le second argument demandé est le WidgetLine à utiliser pour relier ces deux SkillNodes

Num Skills Required est le nombre total de skills que le joueur doit avoir avant de pouvoir débloquer ce SkillNode.

Threshold est un chiffre représentant un palier que ce nœud débloque. Lorsqu'un nouveau palier est atteint, il va activer le Threshold Skill contenu dans le SkillTree. Si un autre skill avec le même palier est activé, le Threshold Skill ne sera pas activé.

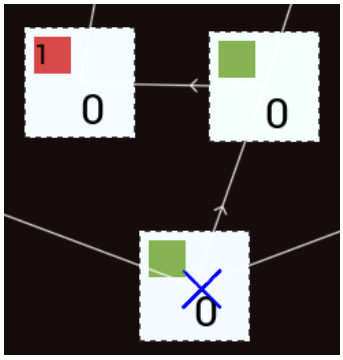



En éditeur, les liens entre les skills ne s'affichent pas automatiquement. Il faut les activer depuis les variables du SkillTreeCanvas

Ceci se fait dans l'onglet Skill Tree Debug.

- Draw Debug permet d'activer ou non la visualisation des liens et des conditions de débloquages de chaques SkillNodes
- Debug Font Size change la taille de la font qui affiche le nombre de skill que le joueur doit posséder pour débloquer le SkillNode
- Debug Font Color change la couleur de la font
- Debug Skill Brush est un Slate Brush permettant d'afficher l'état du skill node, s'il possède un skill ou pas
- Debug Skill Brush Size est la taille du Skill Brush
- Debug Skill Brush Offset est le décalage par rapport au centre du node du Skill Brush
- Debug Has/Has No Skill Color sont les couleurs assignées au Skill Brush s'il possède un Skill ou non

Quand un SkillNode est placé dans le Skill Tree Canvas avec les options de debugs activés, plusieurs informations y sont présentées:




- Le nombre en bas à droite du nœud correspond au nombre de skills nécessaire au déblocage de ce SkillNode.
- Le carré de couleur correspond à l'assignation du Skill au SkillNode. Par défaut, s'il est rouge, aucun skill n'est assigné et s'il est vert un skill a été assigné.
- Le nombre dans le carré de couleur correspond au threshold du SkillNode. Il n'apparaît pas si le SkillNode ne possède pas de threshold.

## SkillDescriptor

Ce widget permet d'afficher les informations d'un skill.

Il nécessite 3 widgets obligatoires:

- Une image "skillImage" qui affiche l'image du skill
- Un text "skillName" qui affiche le nom du skill
- Un text "skillDescription" qui affiche la description du skill

Ce widget n'a besoin de rien d'autre pour fonctionner

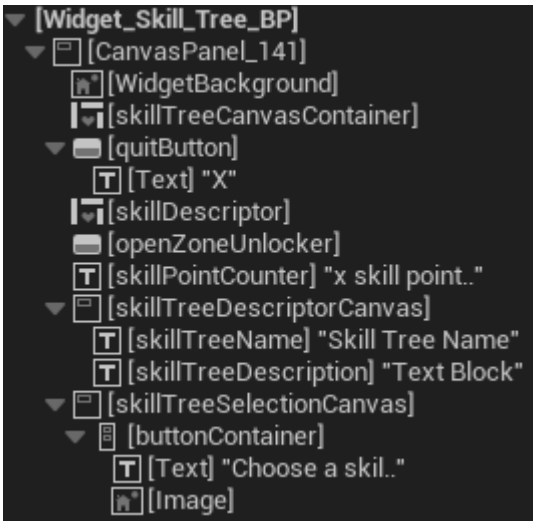
## SkillTree

Ce widget permet la communication et le bon comportement entre les différents widgets composant le skill tree. C'est aussi ce widget qui est affiché en jeu.

Il nécessite 9 widgets obligatoires:

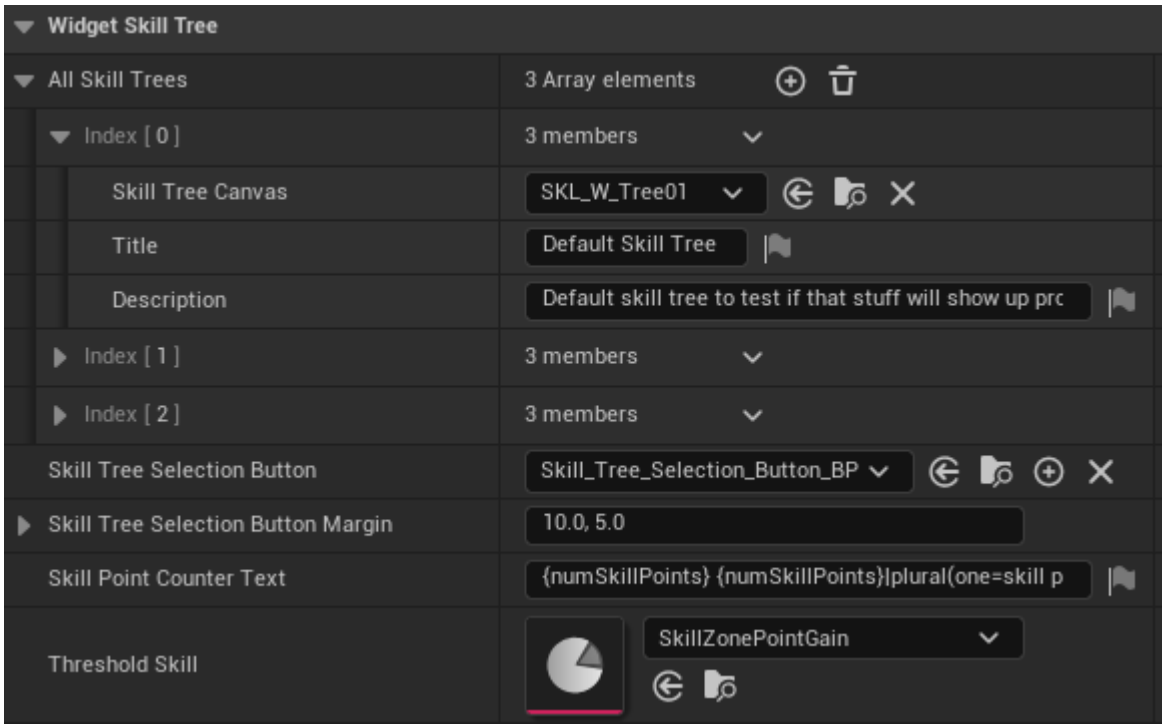
- Un Boutton “quitButton” permettant de quitte le widget
- Un SkillDescriptor “skillDescriptor” qui permet d'afficher les informations d'un skill au survol
- Un SkillTreeContainer “skillTreeCanvasContainer” qui permettera l'affichage d'un skill tree
- Une VerticalBox “buttonContainer” qui contiendra une liste de boutton correspondant au skilltrees que le joueur peut sélectionner
- Un TextBlock “skillPointCounter” qui contient le nombre de skillpoint que le joueur possède
- Un CanvasPanel “skillTreeDescriptorCanvas” qui contient les informations à propos d'un skilltree lorsqu'un boutton est survolé
- Un CanvasPanel “skillTreeSelectionCanvas” qui contient les boutons des différents skilltree sélectionnable
- Un TextBlock “skillTreeName” pour afficher le nom d'un skilltree au survol d'un boutton
- Un TextBlock “skillTreeDescriptor” pour afficher la description d'un skilltree au survol d'un boutton

La plus part de ces widgets peuvent être placé librement, hormis clippingCanvas, skillTreeRoot et skillTreeCanvas qui doivent rester dans cette ordre. Si besoin, voici un ordre qui marche correctement:



En plus de ces widgets, il possède aussi 5 variables :

- All Skill Trees : Un tableau contenant tous les skilltrees que le joueur peut sélectionner, ainsi que ces informations.
  - Skill Tree Canvas : Le BP du SkillTreeCanvas
  - Title : Le nom du skilltree
  - Description : Une description du skilltree
- Skill Tree Selection Button : Le BP des bouttons pour sélectionner le skilltree
- Skill Tree Selection Button Margin : La marge à appliquer autour des bouttons pour sélectionner le skilltree
- Skill Point Counter Text : Le texte à afficher en fonction du nombre de Skillpoints obtenu
- Threshold Skill : un Skill qui sera activé toutes les fois qu'un nouveau palier est atteint dans le SkillTreeCanvas



## *Possibilités du SkillTree*

L'avantage de ce système est le placement libre des SkillNodes, ainsi que la flexibilité de leur condition de déblocage.

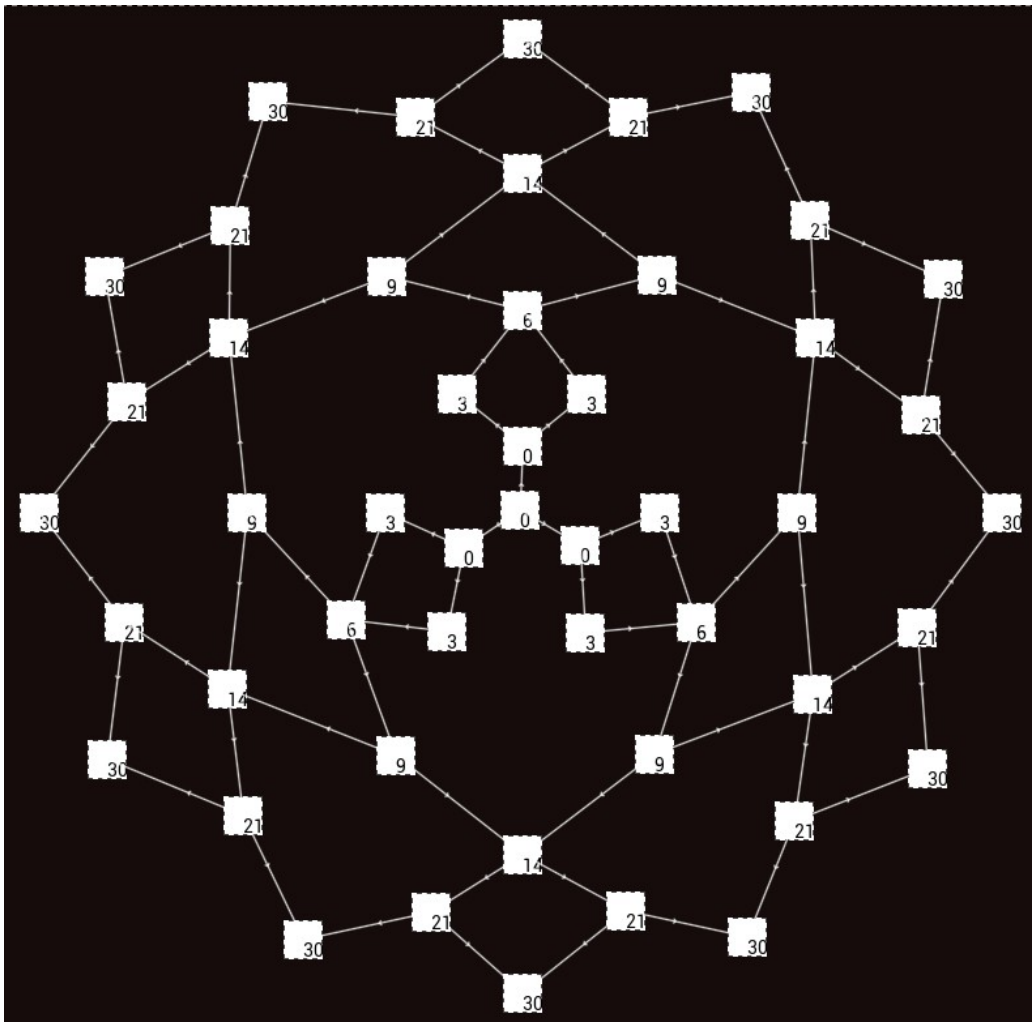
Un SkillNode peut être libre dès le départ, bloqué par un précédent SkillNode, un nombre de skills à obtenir, ou bien même les deux.

Tous les SkillNodes ne doivent pas être obligatoirement reliés. Il est tout à fait possible de créer plusieurs arborescence dans un même SkillTreeCanvas.

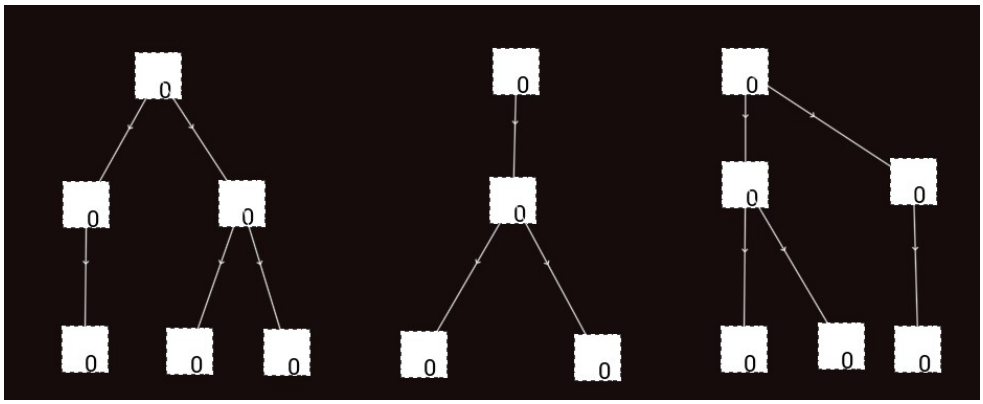
Le SkillTreeCanvas étant un Blueprint séparé du widget principal, il est possible de créer plusieurs SkillTreeCanvas, et de les changer rapidement à des buts de tests.

Voici quelques suggestions de présentations de SkillTrees :

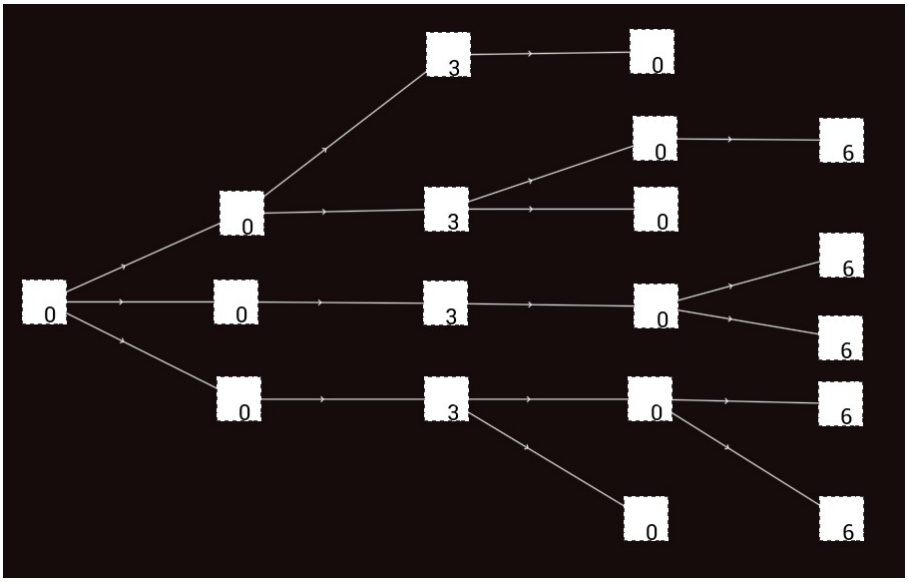
Skill Tree radial



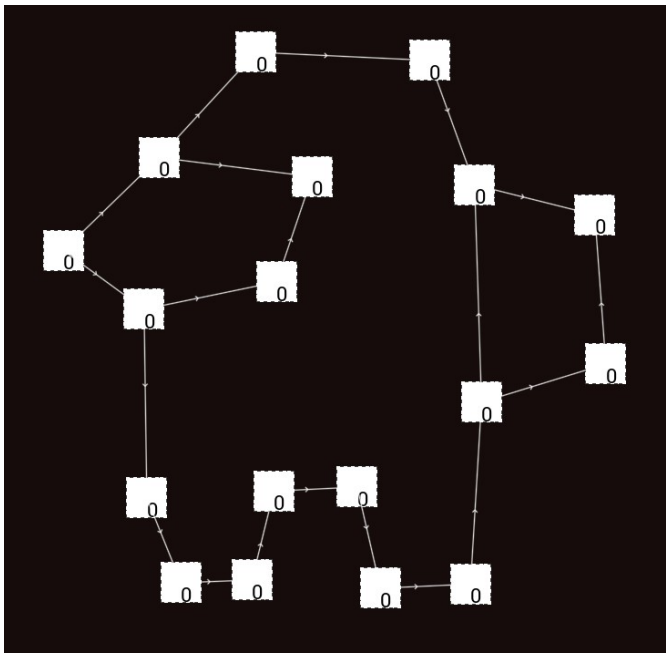
Skill Tree subdivisé



Skill Tree horizontal avec paliers



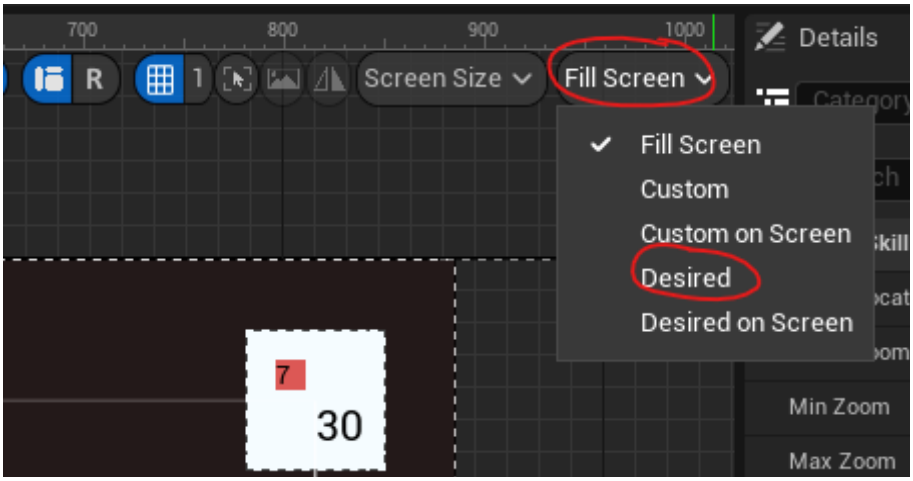
Sus Tree



## *Potentiels problèmes*

### En éditeur, les nœuds disparaissent quand je me déplace

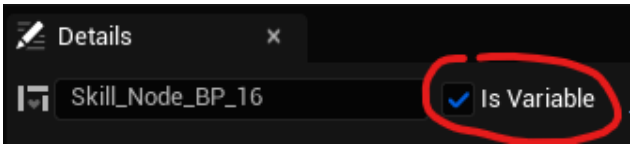
Pour afficher un widget à l'écran, Unreal vérifie d'abbord s'il est bien à l'écran. S'il est hors de l'écran, Unreal le cache. Dans ce cas, c'est probablement le canvas parent de tous les SkillNodes qui passe hors de l'écran, qui du coup cache tout. Pour remédier à cela, il faut agrandir la taille du widget dans l'éditeur. En haut à droite de l'éditeur, changer "Fill Screen" ou l'option activé à ce moment en "Desired"




Entre autre, cette option permet aussi de savoir la taille désirée du SkillTree, ce qui peut être pratique pour savoir la quantité de scroll pour aller d'un bout du SkillTree à l'autre en fonction de la résolution de l'écran du joueur.

### Un/Des SkillNodes n'apparaissent pas dans la liste des précédents skills

Pour qu'un SkillNode apparaisse dans cette liste, il doit être présent dans le widget et être marqué comme étant une variable. Vérifier bien que le SkillNode voulu soit présent dans l'arborescence du widget, ainsi que la case "Is Variable" en haut à droite soit bien cochée.

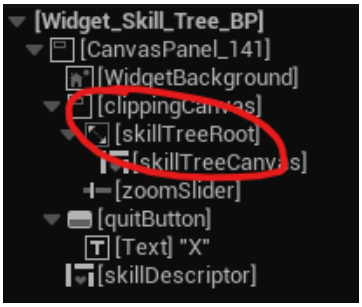


## Le SkillTree en jeu est plus petit/gros qu'en éditeur

Il y a deux potentielles raisons à ce problème.

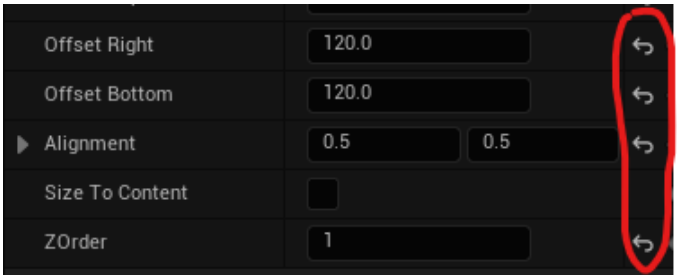
Le premier est que l'éditeur peut être zoomer/dézoomer. Pour vérifier cela, il suffit de regarder en haut à gauche de l'éditeur. Si la valeur est différente de "Zoom 1:1", c'est que vous avez zoomer/dézoomer l'éditeur.


Sinon, cela peut être du au skillTreeRoot posédant une valeur de scale. Pour vérifier cela, aller dans le WidgetSkillTree, sélectionner skillTreeRoot et chercher la valeur "User Specified Scale". Si la valeur est différent de 1, changer là à 1.




## J'ai changé le Blueprint du SkillNode, mais rien n'a changé dans le SkillTree

Malheureusement, c'est comme ça que Unreal fonctionne. Si vous changer le Blueprint d'un widget, ces changements ne seront pas répercutés sur ses différentes instances dans les autres widgets. Le seul moyen d'y remédier et de passer dans le SkillTreeCanvas, sélectionner tous les SkillNodes, retrouver la variables que vous avez changé dans le Blueprint, puis de cliquer sur la flèche à droite pour réinitialiser la valeur à celle dans le Blueprint




## En jeu, les lignes s'affichent au-dessus des skills

Les lignes sont générées au début du jeu, et par conséquent apparaissent par défaut au-dessus des autres widgets. Pour fixer ce problèmes, il suffit de sélectionner tous les SkillNodes, et de changer la variable “Z-Index” à 1 ou une valeur supérieur.