3 Scènes, noeuds, accès aux variables et aux fonctions

3.1 Sauvegarder-réutiliser un noeud

On réutilise les noeuds des sections précédentes ; dans un premier projet, on sauvegarde les noeuds en tant que scènes et on les réutilise dans un second projet.

Fig. 99 présente l'arborescence de la scène avec trois symboles clap de cinéma ; ces symboles clap indiquent des scènes indépendantes dont les modifications s'appliquent à toutes leurs instances dans le projet.

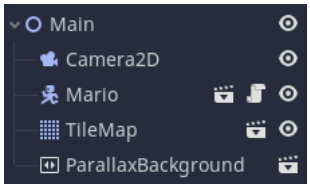

Fig. 99 – Arborescence.

Pour réaliser cette interface, suivre la procédure ci-dessous :

- Dans le premier projet, sauvegarder le noeud `Mario` dans une scène (*click-droit* sur le noeud, puis sélectionner « *Sauvegarder la branche comme scène* »)
  [L'opération inverse est « *Rendre local* »]
  [Le nom du fichier de sauvegarde est `Mario.tscn`]
  [Un symbole de clap de cinéma apparaît sur la droite du noeud `Mario`]
- Sauvergarder le noeud `Tilemap` dans une scène `Tilemap.tscn` et le noeud `ParallaxBackground` dans une scène `ParallaxBackground.tscn`
- Copier dans un second projet, les fichiers `tscn` et les ressources associées
  [Ajouter si nécessaire des touches clavier dans les paramètres du second projet]
- Sélectionner le noeud `Main` du nouveau projet
- Cliquer sur le symbole de lien et sélectionner le fichier `Mario.tscn` pour instancier un nouveau noeud `Mario` de type *KinematicBody2D*
- Instancier les noeuds `Tilemap` et `ParallaxBackground`
  [On retrouve par défaut la première *tilemap* définie dans le fichier `.tscn`]
- Ajouter un noeud `Camera2D` et activer le champ *Current*
- Exécuter permet d'obtenir la Fig. 100

Fig. 100 – Position initiale.

3.2 Modifier la valeur d'une variable d'un noeud parent

On utilise deux scènes : une scène principale `Main.tscn` et une scène `Other.tscn` ; dans la première, on a un noeud `Main` de type *Node2D* et deux noeuds de type *Timer* ; toutes les secondes, on affiche la variable `Var` avec la fonction *print* ; après cinq secondes, on instancie un noeud `other` dans la scène `Main` ; à son insertion dans l'arborescence des noeuds de la scène `Main`, `other` modifie la variable `Var` du `Main`.

Fig. 101 présente l'arborescence de `Main.tscn` ; Fig. 102 présente les fonctions et les variables de `Main.vs`.

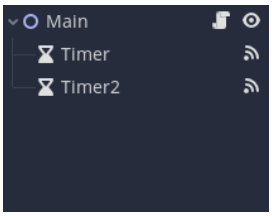

Fig. 101 – Arborescence.

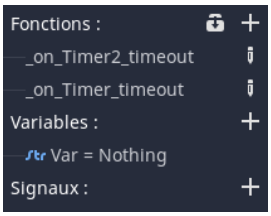

Fig. 102 – Fonctions et variables.

Fig. 103 présente l'arborescence de `Other.tscn` ; Fig. 104 présente les fonctions et les variables de `Other.vs`.

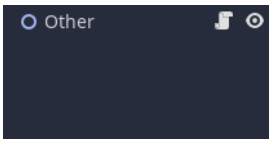

Fig. 103 – Arborescence.

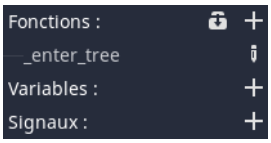

Fig. 104 – Fonctions et variables.

Pour définir un tel processus, suivre la procédure ci-dessous :

— Dans la première scène `Main.tscn`, définir un noeud racine `Main`
— Associer au noeud `Main` un *visual script* `main.vs`
— Définir une variable `Var` de type *String*
  [`Var` est initialisée à la valeur « Nothing »]
— Ajouter un noeud `Timer` de type *Timer*
  [Le champ *Wait Time* est à 1]
  [Le champ *Autostart* est activé]
— Connecter une fonction *_on_Timer_timeout* vers `main.vs`
— Ajouter un noeud `Timer2` de type *Timer*
  [Le champ *Wait Time* est à 5]
  [Le champ *One Shot* est activé]
  [Le champ *Autostart* est activé]
— Connecter une fonction *_on_Timer2_timeout* vers `main.vs`
— Dans la deuxième scène `Other.tscn`, définir un noeud racine `Other`
— Associer au noeud `Other` un *visual script* `other.vs`

— Ajouter la fonction prédéfinie *_enter_tree* dans `other.vs`
[La séquence exécutée à l'appel de cette fonction est présentée en Fig. 105]
[Pour obtenir la fonction *.set* qui est en fait *Node2D.set*]
[Ajouter la fonction dont le prototype est *Set(String,Var)*]
[En utilisant l'inspecteur, modifier le champ *Call Mode* de *Self* à *Instance*]
[Associer le retour de *get_parent* au paramètre *instance*]
[Ecrire « Var » dans le champ du paramètre *property*]
[Associer la constante `Hello` de type *String* au paramètre *value*]
[On revient maintenant dans la première scène `Main.tscn`]
— Ajouter les fonctions *print* et `Get Var`
— Charger la scène `Other.tscn` par glisser-déposer dans `Main.vs`
[Ajouter l'appel à *PackedScene.instance* en sélectionnant l'objet `Fireball.tscn`]
[Placer le noeud créé dans l'arborescence avec *add_child*]
[Relier le retour de *PackedScene.instance* avec le paramètre *node*] comme le présente la Fig. 106
— Exécuter et constater dans le terminal l'affichage de *Nothing* et son remplacement par *Hello* après cinq secondes
[`Var` est affichée toutes les secondes]

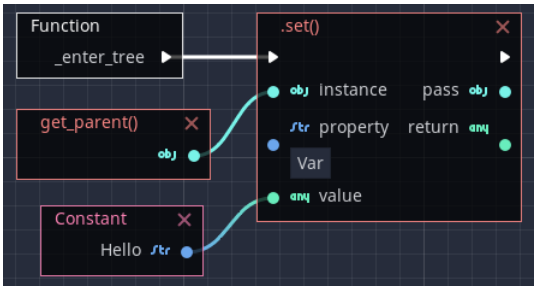

Fig. 105 – `Other.vs`.

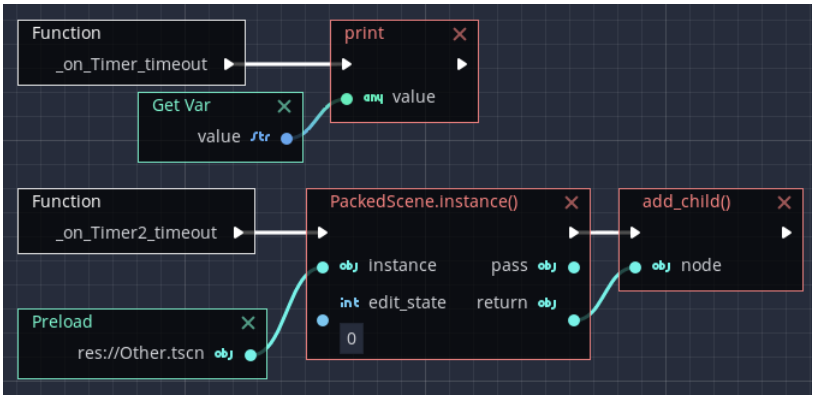

Fig. 106 – `Main.vs`.

3.3 Appeler une fonction d'une autre scène

On utilise deux scènes `Main.tscn` et `Other.tscn` comme dans la section précédente ; on ajoute dans `Main.vs` une fonction *update_var* pour modifier `Var`.

Les arborescences sont identiques à la section précédente ; Fig. 107 présente les fonctions et les variables de `Main.vs`.

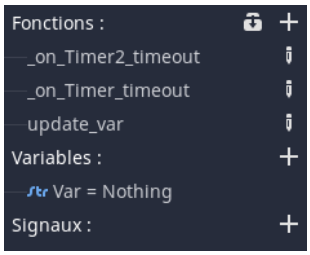

Fig. 107 – Fonctions et variables de `Main.vs`.

Pour définir un tel processus, suivre la procédure ci-dessous :

- Dans la première scène `Main.tscn`, définir un noeud racine `Main`
- Associer au noeud `Main` un *visual script* `main.vs`
- Définir une variable `Var` de type *String* initialisée à « Nothing »
- Ajouter un noeud `Timer` de type *Timer* avec les mêmes paramètres que la section précédente et connecter *_on_Timer_timeout* vers `main.vs`
- Ajouter un noeud `Timer2` de type *Timer* avec les mêmes paramètres que la section précédente et connecter *_on_Timer2_timeout* vers `main.vs`
- Ajouter la fonction `update_var` dans `main.vs` comme présenté en Fig. 108
- Charger la scène `Other.tscn` par glisser-déposer dans `Main.vs` et ajouter l'appel à *PackedScene.instance*
- Ajouter le noeud dans l'arborescence avec *add_child*

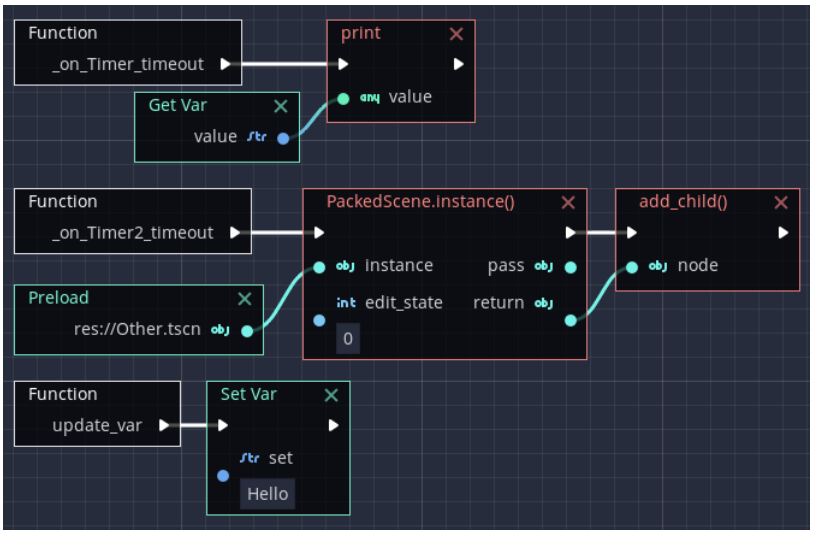

Fig. 108 – `Main.vs`.

— Dans la deuxième scène `Other.tscn`, définir un noeud racine `Other`
— Associer au noeud `Other` un *visual script* `other.vs`
— Ajouter la fonction prédéfinie *_enter_tree* dans `other.vs`
[La séquence exécutée à l'appel de cette fonction est présentée en Fig. 109]
[Pour obtenir l'appel de fonction *Node2D.update_var* dans `Other.vs`]
[Ajouter la fonction dont le prototype est *Rotate(Float)* de la classe *Node2D*]
[En utilisant l'inspecteur, modifier le champ *Call Mode* de *Self* à *Instance*]
[Associer le champ *Base Script* au fichier `Main.vs`]
[Modifier le champ *Function* à *update_var* dont le prototype est dans les *Script Methods*]
[Associer le retour de *get_parent* au paramètre *instance*]
— Exécuter et constater dans le terminal l'affichage de *Nothing* et son remplacement par *Hello* après cinq secondes

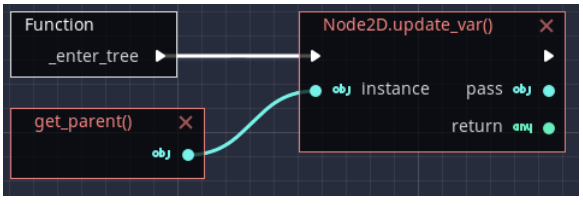

Fig. 109 – `Other.vs`.

Il est important de noter que l'appel d'une fonction d'une instance d'une scène implique de connaître la position relative de l'instance par rapport au noeud courant dans l'arborescence ; l'inspecteur permet une fois de plus de créer des fonctions plus paramétrables.

3.4 Appeler une fonction avec paramètres d'une autre scène

On utilise deux scènes `Main.tscn` et `Other.tscn` comme dans la section précédente ; on ajoute dans `Main.vs` une fonction *update_var* avec un paramètre de type *String* pour modifier `Var`.

Les arborescences sont identiques à la section précédente ; dans la liste des fonctions et des variables, la fonction *update_var* attend un argument de type *String* mais cela n'apparaît que dans son utilisation dans les *visual script* `Main.vs` pour sa déclaration et `Other.vs` pour son utilisation.

Pour définir un tel processus, suivre la procédure ci-dessous :

- Dans la scène `Main.tscn`, définir un noeud racine `Main`
- Associer un *visual script* `main.vs`
- Définir une variable `Var` de type *String* initialisée à « Nothing »
- Ajouter un noeud `Timer` de type *Timer* avec les mêmes paramètres que la section précédente et connecter *_on_Timer_timeout* vers `main.vs`
- Ajouter un noeud `Timer2` de type *Timer* avec les mêmes paramètres que la section précédente et connecter *_on_Timer2_timeout* vers `main.vs`
- Ajouter la fonction `update_var` dans `main.vs` comme présenté en Fig. 110
  [Pour obtenir une fonction avec paramètre, écrire 1 dans le champ *Argument count* avec l'inspecteur]
  [Un nouveau champ *Argument 1* apparaît sous *Argument count*]
  [Définir le type de *Argument 1* comme étant *String*]
- Ajouter un appel à *Set Var* en associant *arg1* à *set*
- Instancier la scène `Other.tscn` dans la fonction *_on_Timer2_timeout*
- Ajouter l'instance dans l'arborescence avec *add_child*

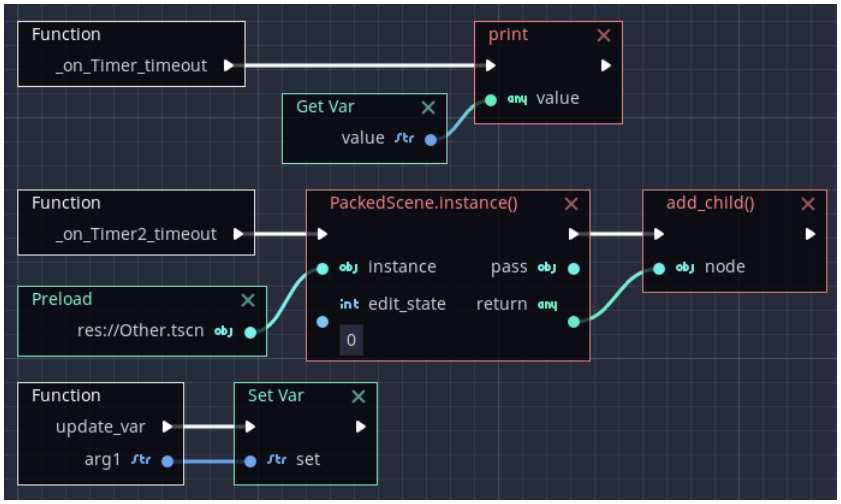

Fig. 110 – `Main.vs`.

— Dans la deuxième scène `Other.tscn`, définir un noeud racine `Other`
— Associer au noeud `Other` un *visual script* `other.vs`
— Ajouter la fonction prédéfinie *_enter_tree* dans `other.vs`
[La séquence exécutée à l'appel de cette fonction est présentée en Fig. 111]
[Pour obtenir l'appel de fonction *Node2D.update_var* dans `Other.vs`]
[Ajouter la fonction dont le prototype est *Rotate(Float)* de la classe *Node2D*]
[En utilisant l'inspecteur, modifier le champ *Call Mode* de *Self* à *Instance*]
[Associer le champ *Base Script* au fichier `Main.vs`]
[Modifier le champ *Function* à *update_var* dont le prototype est dans les *Script Methods*]
[Associer le retour de *get_parent* au paramètre *instance*]
[Associer la constante `Hello` de type *String* au paramètre *arg1*]
— Exécuter et constater dans le terminal l'affichage de *Nothing* et son remplacement par *Hello* après cinq secondes

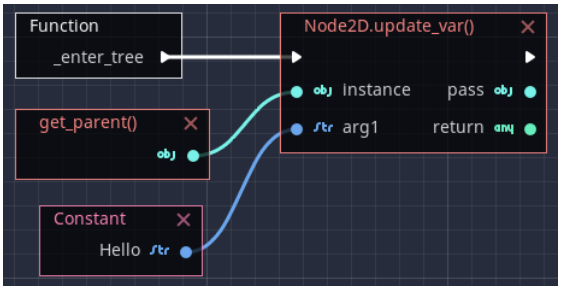

Fig. 111 – `Other.vs`.

3.5 Appeler une fonction retournant une valeur d'une autre scène

On utilise deux scènes : une scène principale `Main.tscn` et une scène `Other.tscn` ; dans la première, on a un noeud `Main` de type Node2D et un noeud de type *Timer* ; dans `Main`, on instancie dynamiquement (*i.e.* lors de l'exécution avec la fonction *PackedScene.instance*) un noeud `Other` ; dans `Main` toujours, toutes les secondes, on affiche une valeur `count` de `Other` retournée par une fonction de `Other` ; dans la scène `Other`, on a un noeud `Other` de type Node2D et un noeud de type *Timer* ; toutes les deux secondes, on modifie la valeur retournée par la fonction appelée par `Main`.

Fig. 112 et 113 présentent l'arborescence de `Main.tscn` et de `Other.tscn`.

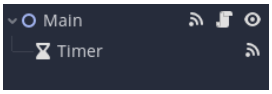

Fig. 112 – Arborescence de `Main`.

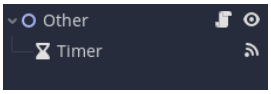

Fig. 113 – Arborescence de `Other`.

Fig. 114 et 115 présentent les fonctions et les variables de `Main.tscn` et de `Other.tscn`.

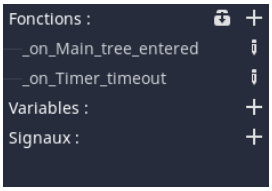

Fig. 114 – Fonctions de `Main`.

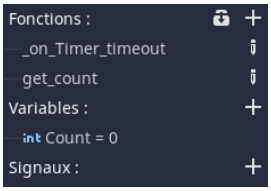

Fig. 115 – Fonctions de `Other`.

Pour définir un tel processus, suivre la procédure ci-dessous :

- Dans la scène `Main.tscn`, définir un noeud racine `Main`
- Associer un *visual script* `main.vs`
- Ajouter un noeud `Timer` de type *Timer*
  [Le champ *Wait Time* est à 1]
  [Le champ *Autostart* est activé]
- Connecter une fonction *_on_Timer_timeout* vers `main.vs`
- Ajouter la fonction *_on_Main_tree_entered* dans `main.vs` comme présenté en Fig. 116
- Dans la scène `Other.tscn`, définir un noeud racine `Other`
- Associer un *visual script* `other.vs`
- Ajouter un noeud `Timer` de type *Timer*
  [Le champ *Wait Time* est à 2]
  [Le champ *Autostart* est activé]
- Connecter une fonction *_on_Timer_timeout* vers `other.vs`
- Définir la séquence associée à *_on_Timer_timeout* comme présenté en Fig. 117
- Ajouter une variable `Count` de type *int*
- Ajouter la fonction *get_Count* qui retourne la valeur courante de `Count` comme présenté en Fig. 117

— Exécuter et constater dans le terminal l'affichage d'un compteur en affichant chaque valeur deux fois (*i.e.* 0,0,1,1,2,2, ...)

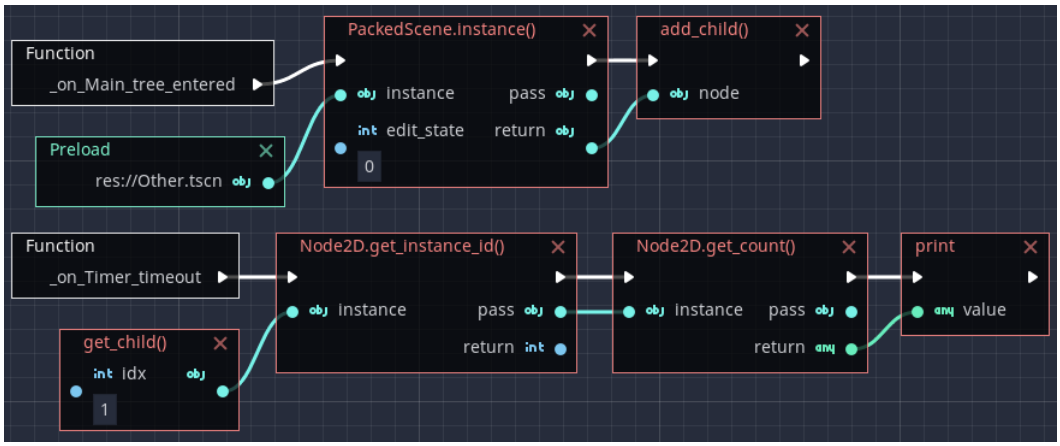

Fig. 116 – `Main.vs`.

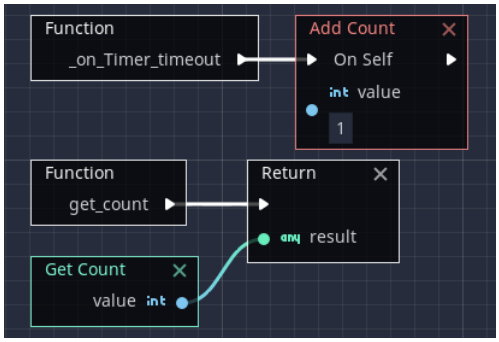

Fig. 117 – `Other.vs`.

3.6 Afficher les éléments d'un tableau avec un *for*

On utilise une scène `Main.tscn` avec une variable `T` de type *Array* de taille 3 ; les valeurs de `T` sont 10,11 et 12 ; on affiche le contenu de `T` avec une boucle *for*.

Fig. 118 présente l'arborescence de `Main.tscn` ; Fig. 119 présente les fonctions et les variables de `Main.vs`.

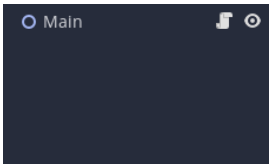

Fig. 118 – Arborescence.

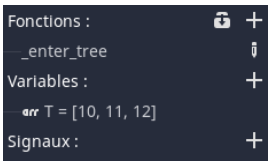

Fig. 119 – Fonctions et variables.

Pour définir un tel processus, suivre la procédure ci-dessous :

- Dans la scène `Main.tscn`, définir un noeud racine `Main`
- Associer un *visual script* `main.vs`
- Ajouter une variable `T` de type *Array*
  [Ajouter trois valeurs 10,11 et 12 de type *int*]
- Ajouter la fonction *_enter_tree* dans `main.vs`
  [Au chargement d'un fichier `.tscn`, l'arbre de la scène se construit en commençant par la racine ; quand un noeud est construit, la fonction *_init* est appelée ; quand un noeud est ajouté dans l'arbre, la fonction *_enter_tree* est appelée ; quand tous les enfants d'un noeud ont été ajoutés et sont prêts à être utilisés, la fonction *_ready* est appelée ; quand un noeud est retiré de l'arbre de la scène, la fonction *_exit_tree* est appelée ; pour les fonctions *_enter_tree*, *_ready* et *_exit_tree*, il est équivalent d'utiliser les signaux *ready*, *tree_entered* et *tree_exited* ; le signal *tree_exiting* permet de réaliser des opérations avant la sortie de l'arbre.]
- Définir la séquence associée à *_enter_tree* comme présenté en Fig. 120
  [Pour chaque élément, on appelle la fonction *print* avec le lien blanc partant de *each* dans le composant *Iterator* ; à chaque appel de *print*, on passe en argument la valeur de l'élément courant avec le lien vert partant de *elem*]
- Exécuter et constater dans le terminal l'affichage des valeurs de `T`

Fig. 120 – `Main.vs`.

3.7 Afficher les éléments d'un tableau sans *for*

On utilise une scène `Main.tscn` avec une variable `T` de type *Array* de taille 3 ; les valeurs de `T` sont 10,11 et 12 ; on ajoute à la scène `Main.tscn` une variable `N` de type *int* de valeur 3 ; on décrémente la valeur de `T` tant qu'elle est strictement positive, on retire la première valeur de `T` et on affiche cette valeur ; à la différence de la solution précédente, on vide `T` de ces éléments au cours de l'exécution.

Fig. 121 présente l'arborescence de `Main.tscn` ; Fig. 122 présente les fonctions et les variables de `Main.vs`.

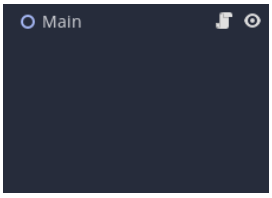

Fig. 121 – Arborescence.

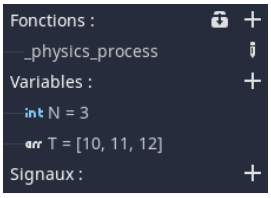

Fig. 122 – Fonctions et variables.

Pour définir un tel processus, suivre la procédure ci-dessous :

- Dans la scène `Main.tscn`, définir un noeud racine `Main`
- Associer un *visual script* `main.vs`
- Ajouter une variable `T` de type *Array*
  [Ajouter trois valeurs 10,11 et 12 de type *int*]
- Ajouter une variable `N` de type *int* et de valeur 3
- Ajouter la fonction *_physics_process* dans `main.vs`
- Définir la séquence associée à *_physics_process* comme présenté en Fig. 123
  [*Array.pop_front* supprime le premier élément de `T` et retourne cet élément]
  [L'élément retourné est affiché par la fonction *print*]
  [Après trois appels de la fonction *_physics_process*, la condition n'est plus vérifiée et la séquence des fonctions *Subtract*, *Array.pop_front* et *print* n'est plus exécutée]
- Exécuter et constater dans le terminal l'affichage des valeurs de `T`

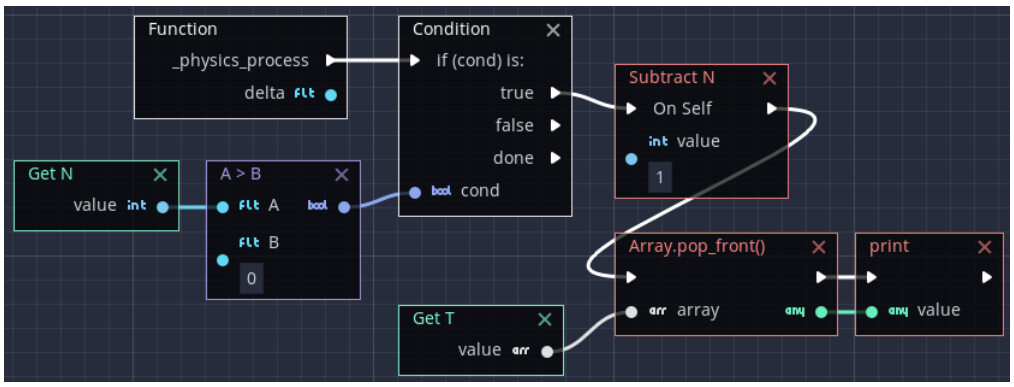

Fig. 123 – `Main.vs`.

3.8 Afficher l'interpolation linéaire d'un *Vector2*

On utilise une scène Main.tscn avec un *Vector2* `A` égal à (100,200) ; on décrémente une valeur de `N` initialisée à 5 ; tant que `N` est strictement positive, on calcule l'interpolation vers (50,50) et on met à jour `A` avec cette valeur.

Fig. 124 présente l'arborescence de `Main.tscn` ; Fig. 125 présente les fonctions et les variables de `Main.vs`.

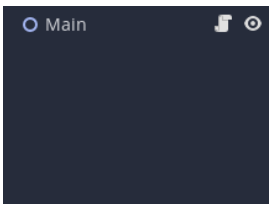

Fig. 124 – Arborescence.

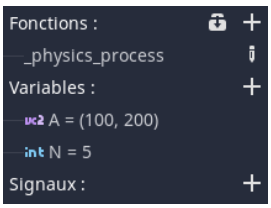

Fig. 125 – Fonctions et variables.

Pour définir un tel processus, suivre la procédure ci-dessous :

- Dans la scène `Main.tscn`, définir un noeud racine `Main`
- Associer un *visual script* `main.vs`
- Ajouter une variable `A` de type *Vector2* égale à (100,200)
- Ajouter une variable `N` de type *int* et de valeur 5
- Ajouter la fonction *_physics_process* dans `main.vs`
- Définir la séquence associée à *_physics_process* comme présenté en Fig. 126
  [on affiche les valeurs d'interpolation de (100,200) vers (50,50)]
  [Après cinq appels de la fonction *_physics_process*, la condition n'est plus vérifiée et l'interpolation vers (50,50) s'arrête]
  [Avec un coefficient de 0.2, la première valeur est $(100 \times 0.8 + 50 \times 0.2, 200 \times 0.8 + 50 \times 0.2)$, soit (90,170)]
- Exécuter et constater dans le terminal l'affichage des couples (90,170), (82,146), (76,127), (70,111) et (66,99)

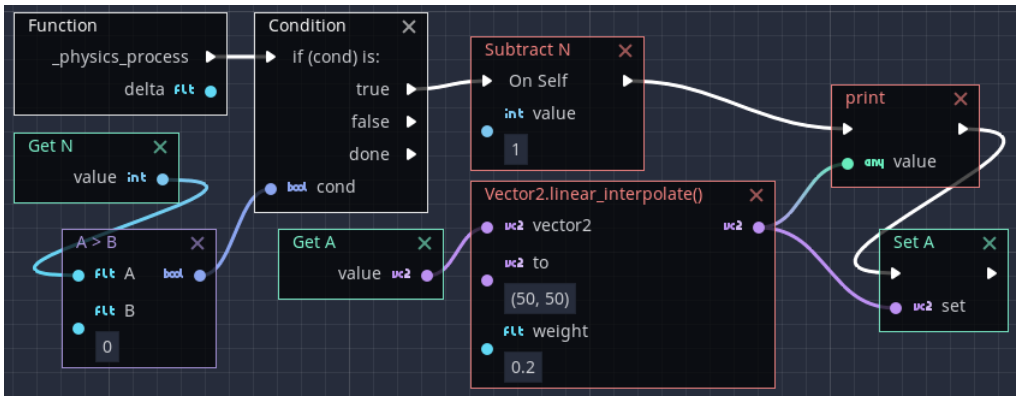

Fig. 126 – `Main.vs`.