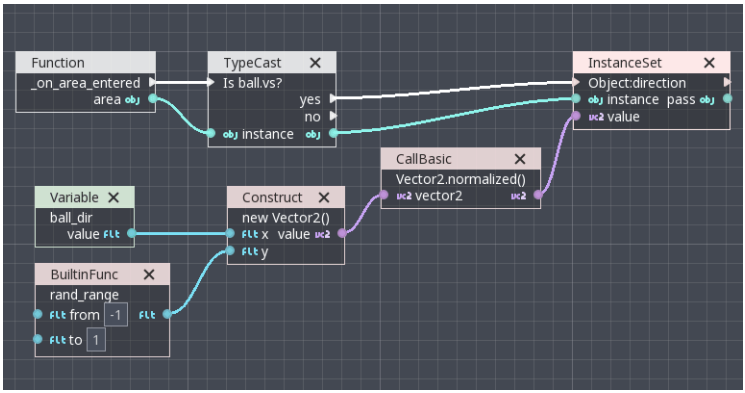
4

Initiation à la création de scripts avec Godot

Comme tous les moteurs de jeux, Godot permet la création de scripts. L'écriture de scripts permet de mettre en place des interactions entre le joueur et le jeu. Nous pourrons récupérer les *inputs* (comme les entrées clavier/souris), gérer le comportement des objets (comme le déplacement du personnage ou des éléments à collecter) ou encore mettre en place les conditions de réussite/échec du jeu.

La création de scripts passe par de la programmation. Le langage de programmation principal de Godot est le GDScript, une sorte de Python. Ce langage est assez simple à prendre en main pour les débutants tout en offrant une grande puissance. Depuis la version 3 de Godot, d'autres langages de programmation ont été ajoutés à la liste des langages supportés par le logiciel, notamment le C# et un langage en *visual scripting* permettant de coder de façon visuelle par assemblage de blocs.

Figure 4.1 : *Le visual scripting sous Godot*



Bien que ces autres langages soient attrayants, nous n'utiliserons dans ce livre que le GDScript et c'est le langage que je vous conseille d'utiliser sous Godot. Il est préférable d'utiliser GDScript d'une part car c'est le langage officiel depuis le début et celui qui

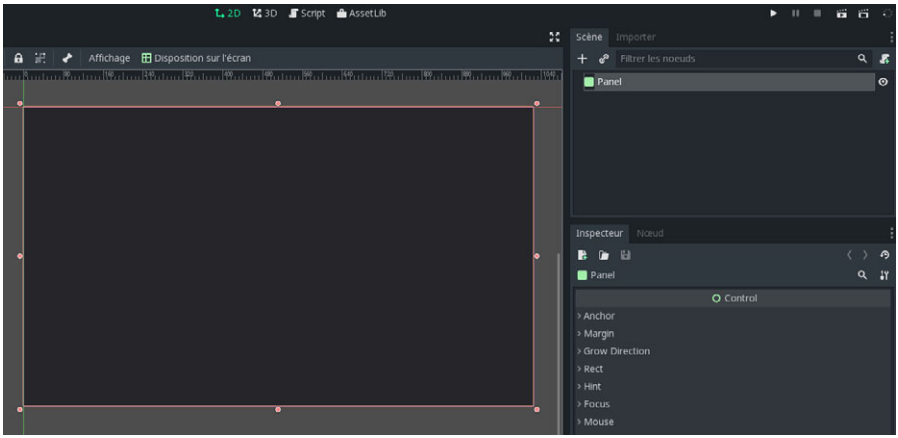
est le plus utilisé par la communauté et donc pour lequel vous trouverez le plus de documentation ; d'autre part car l'intégration de ces nouveaux langages n'est pas encore suffisamment mature comparée à celle du GDScript.

Le GDScript est puissant, performant, dispose d'un éditeur intégré à Godot avec de la coloration syntaxique, supporte de très nombreuses fonctionnalités et est simple à apprendre. Nous allons découvrir comment écrire nos scripts en GDScript.

4.1. Premier script

Commencez par créer une nouvelle scène vide afin de partir sur une base simple. Créez ensuite un node Panel (Ctrl+A) que vous agrandirez pour couvrir la taille de l'écran de jeu.

Figure 4.2 : *Création d'un Panel*



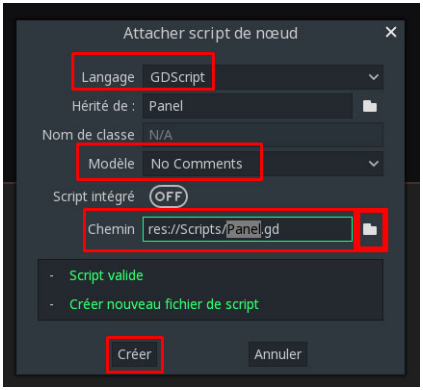
Note *› Le Panel est le composant qui est en général utilisé pour contenir des éléments d'interface comme du texte, des boutons et des champs texte. Un menu principal ou un menu pause sera par exemple conçu dans un panel.*

Nous avons créé un panel car nous avons besoin d'un node principal pour pouvoir créer un script. Nous aurions pu créer n'importe quel autre type de node. Nous avons choisi un Panel car nous allons créer plus tard une UI (User Interface).

Lorsque vous avez un node dans votre scène, vous avez la possibilité d'ajouter un script à ce node. Pour cela, cliquez sur l'icône Script se trouvant en haut à droite du panneau Scène.

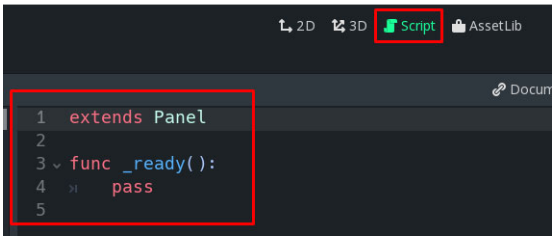
Lors de la création d'un script, vous devez sélectionner le langage ; nous laisserons le GDScript par défaut. Vous devez également choisir le modèle, nous allons sélectionner NO COMMENTS pour avoir un code sans les commentaires par défaut. Nous devons ensuite spécifier un chemin : cliquez sur l'icône en forme de dossier afin de choisir le chemin d'enregistrement (je vous invite à sauvegarder votre script dans un sous-dossier que vous nommerez `Scripts`). Une fois que tout est paramétré, cliquez sur CRÉER.

Figure 4.3 : *Configuration du script*



Lorsque vous validez, le script se crée et s'ouvre comme sur la Figure 4.4.

Figure 4.4 : *Script par défaut*



L'espace de travail SCRIPT s'est automatiquement ouvert. Quelques lignes de code sont déjà écrites par défaut. Pour comprendre, nous allons analyser ce code.

`extends` est un mot clé qui permet d'hériter de propriétés. Ici nous héritons des propriétés de `Panel` (car notre node est un Panel). L'héritage permet à l'enfant (le script) d'accéder

aux propriétés du parent (ici Panel). Cela nous permettra de déclencher des événements propres au Panel si besoin.

Le mot clé `func` est très important. *func* signifie *function* (*fonction* en français). Une fonction est une partie du programme qui exécute des opérations bien précises. Par exemple nous pouvons créer une fonction `soustraire` qui prend en paramètre `nombre1` et `nombre2` et qui retournerait la soustraction de ces deux nombres. Cette fonction pourrait ressembler à cela :

```
func soustraire(nb1, nb2):
  return nb1 - nb2
```

Le mot clé `return` permet de renvoyer une valeur. Une fonction peut être appelée par son nom, dans notre cas ce serait : `soustraire(10,5)`. Le résultat serait alors 5.

La fonction écrite par défaut par Godot est la fonction `_ready()`. Cette fonction est particulière car elle s'exécute au lancement du programme. C'est la première fonction qui est lancée à l'ouverture de la scène.

Une autre fonction importante est la fonction `_process(delta)`. Cette fonction tourne en boucle. Cela signifie que si votre jeu tourne en 60 images par seconde, la fonction s'exécutera 60 fois par seconde. Cela peut être très utile par exemple pour les animations afin de faire avancer le personnage de façon fluide.

`_process` a un paramètre par défaut qui est `delta`. Ce paramètre est en fait le temps écoulé entre deux frames (entre deux images). Ce paramètre permet par exemple de calibrer vos animations afin que leur vitesse soit la même indépendamment de la puissance de l'ordinateur qui exécute le programme.

Enfin, le dernier mot clé que nous voyons est `pass`. Une fonction ne peut pas être vide sinon une erreur se déclenche. `pass` permet de créer des fonctions vides sans que le programme n'indique d'erreur.