

>>> CODING WEEKS  
**LOTREO**

*Legends Of The Eternal Realm:  
Epic Odyssey*

2023-2024

# SOMMAIRE

- 
- 1 **Introduction et motivations**
  - 2 **Environnement Unity, MVC et Git**
  - 3 **Caractérisation des usages et MVPs**
  - 4 **Structuration du code et fonctionnalités**
  - 5 **Conclusion et apprentissages**
  - 6 **LOTREO : Le jeu**

# 1 - Introduction et motivations

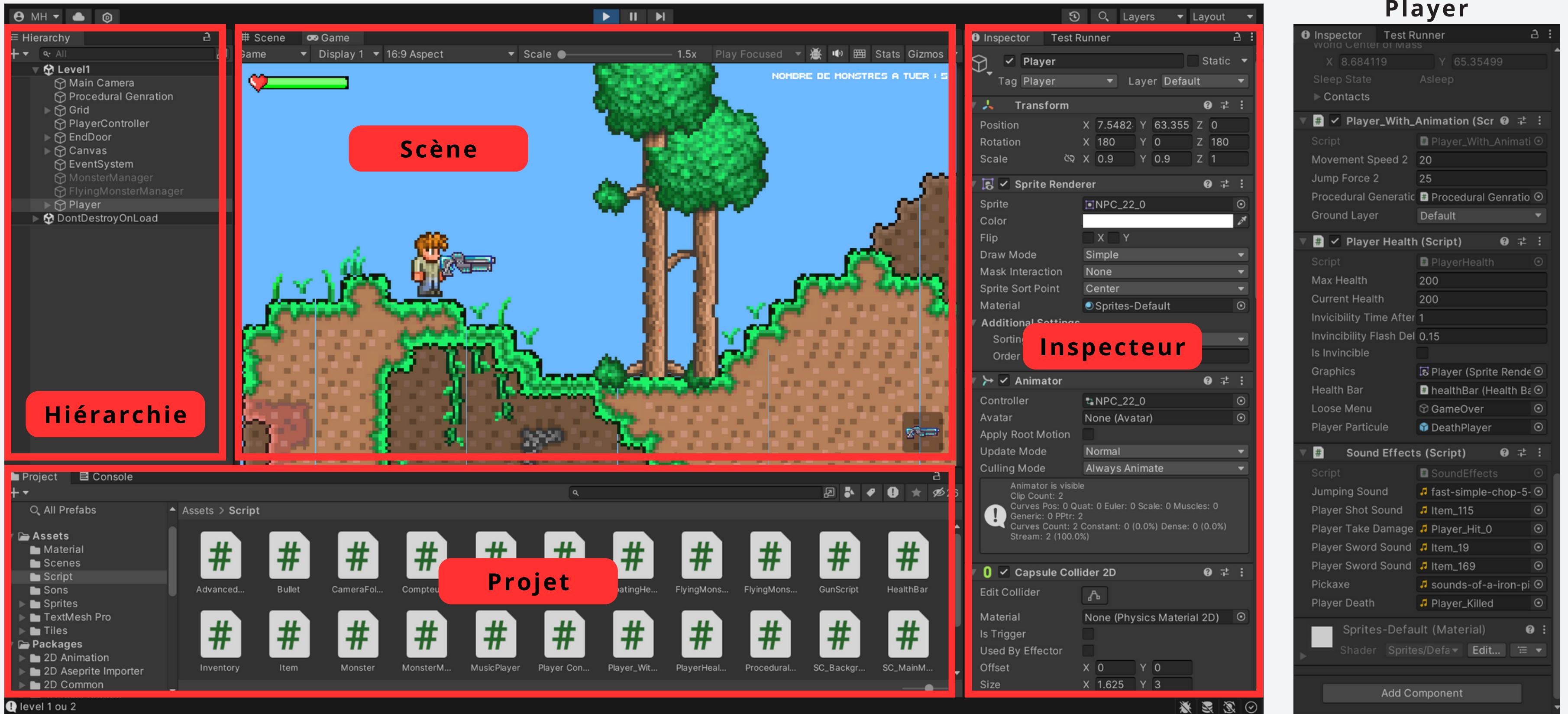
*Pourquoi coder notre jeu sur Unity en C# ?* 

- Découverte de l'écosystème Unity, très robuste et largement utilisé dans le développement de jeux multiplateformes.
- Apprentissage du langage C#. C'est un langage de programmation orienté objet robuste et largement utilisé dans l'industrie du jeu. Il offre des fonctionnalités avancées, une syntaxe claire et une gestion automatique de la mémoire.
- Continuité du projet après les Coding Weeks.

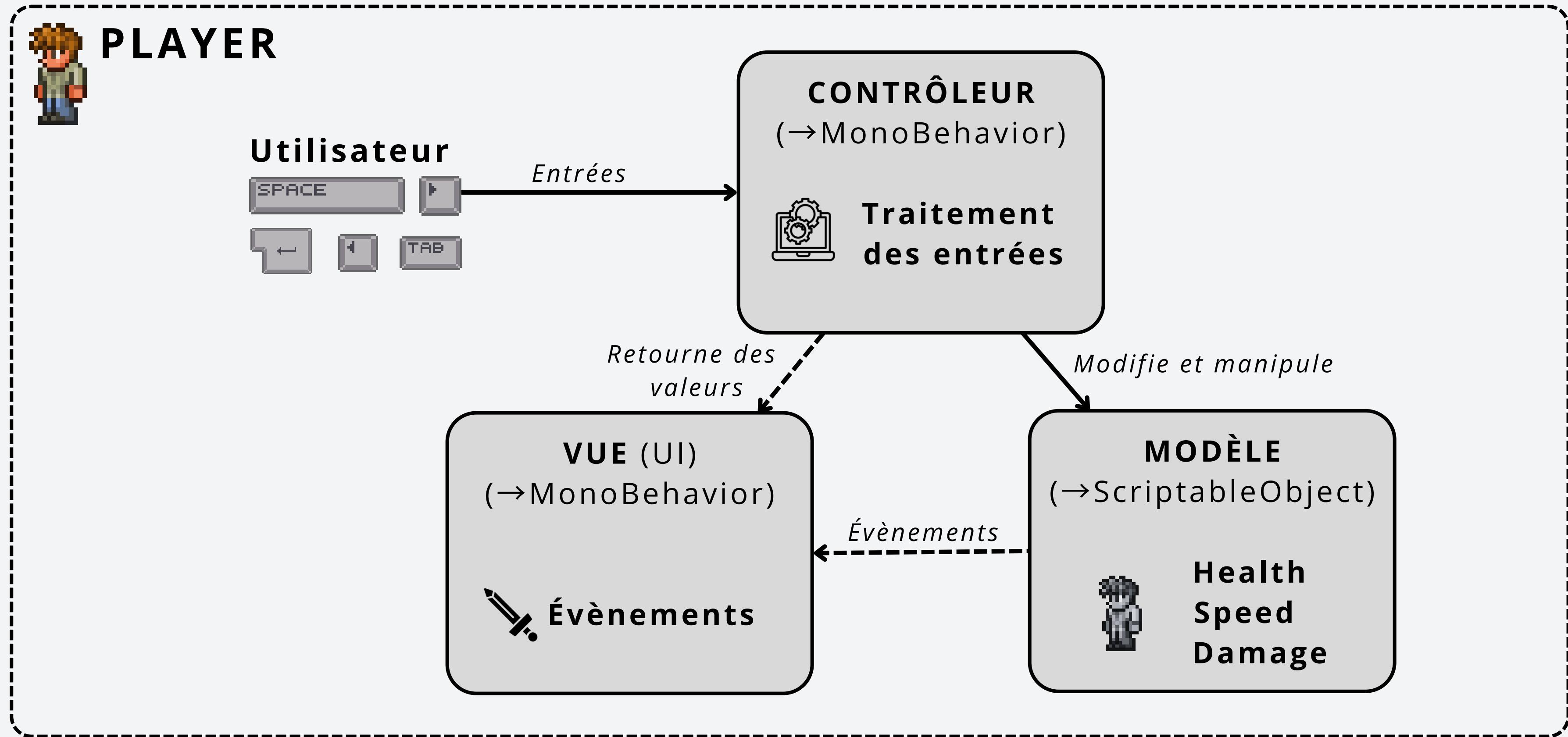
*Quels ont été les apports du projet 2048 et comment la difficulté a-t-elle évoluée ?*

# 2.1 - Environnement Unity

GameObject  
Player



## 2.2 - MVC



## 2.3 - Gestion du Git

Name	Last commit
📁 2D tile map	Correction monstre flying
📌 .gitignore	test_merge gitignore modified
📝 README.md	Update file README.md

***Pourquoi avons-nous un fichier .gitignore sur GitLab ?***

- Omission du cache Unity et VScode lors de chaque merge sur le main.
- Allégement de la taille du jeu mis en ligne sur GitLab (1,4 Go → 24 Mo).

# 3.1 - Caractérisation des usages

- Public cible
- Objectifs du jeu
- Mécanismes de jeu
- Complétude du jeu
- Rejouabilité et caractère captivant
- Univers du jeu



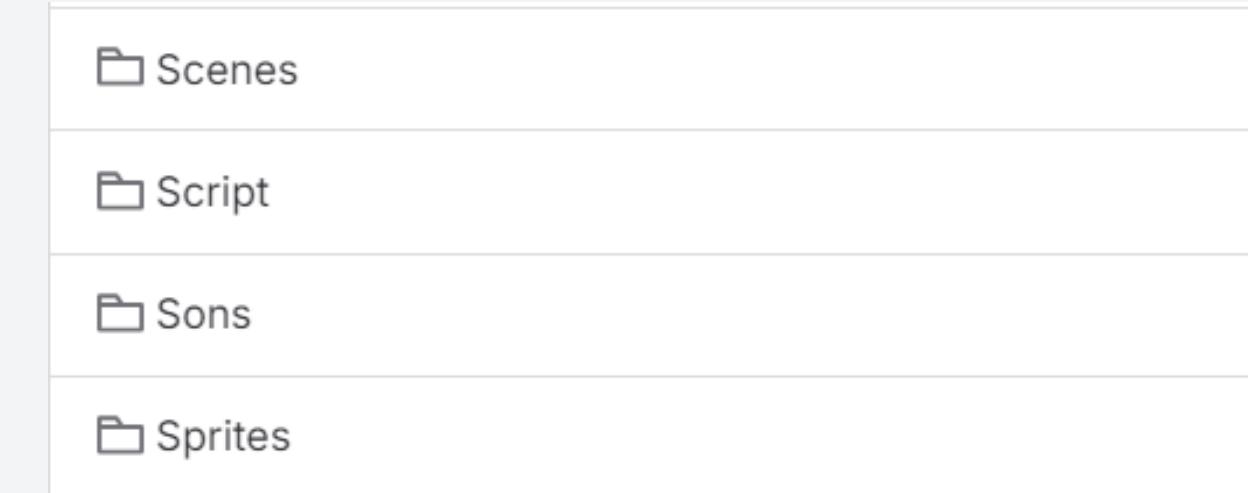
## 3.2 - MVPs



# 4.1 - Structuration du code

## **Gros grain**

- Image (.png) dans Assets/Sprites/
- Fichier de code (.cs) dans Assets/Script/
- Scène de jeu (.unity) dans Assets/Scenes/
- Son du jeu (.mp3) dans Assets/Sons/
- Préfabriqué (.prefab) dans Assets/



## **Petit grain**

- Découpage du code en de nombreux scripts
- Nom de variable explicite (santé du joueur → playerHealth)
  - Paramètres libres pour une plus grande versatilité du projet
- Commentaire (//commentaire)

## **Tester les fonctions**

- Utilisation de l'interface graphique, permettant des ajustements instantanés pour perfectionner chaque détail.
- Recours aux "Debut.Log(\*Message\*)" (équivalent du print() en Python)
- Bêta tests des fonctions en groupe

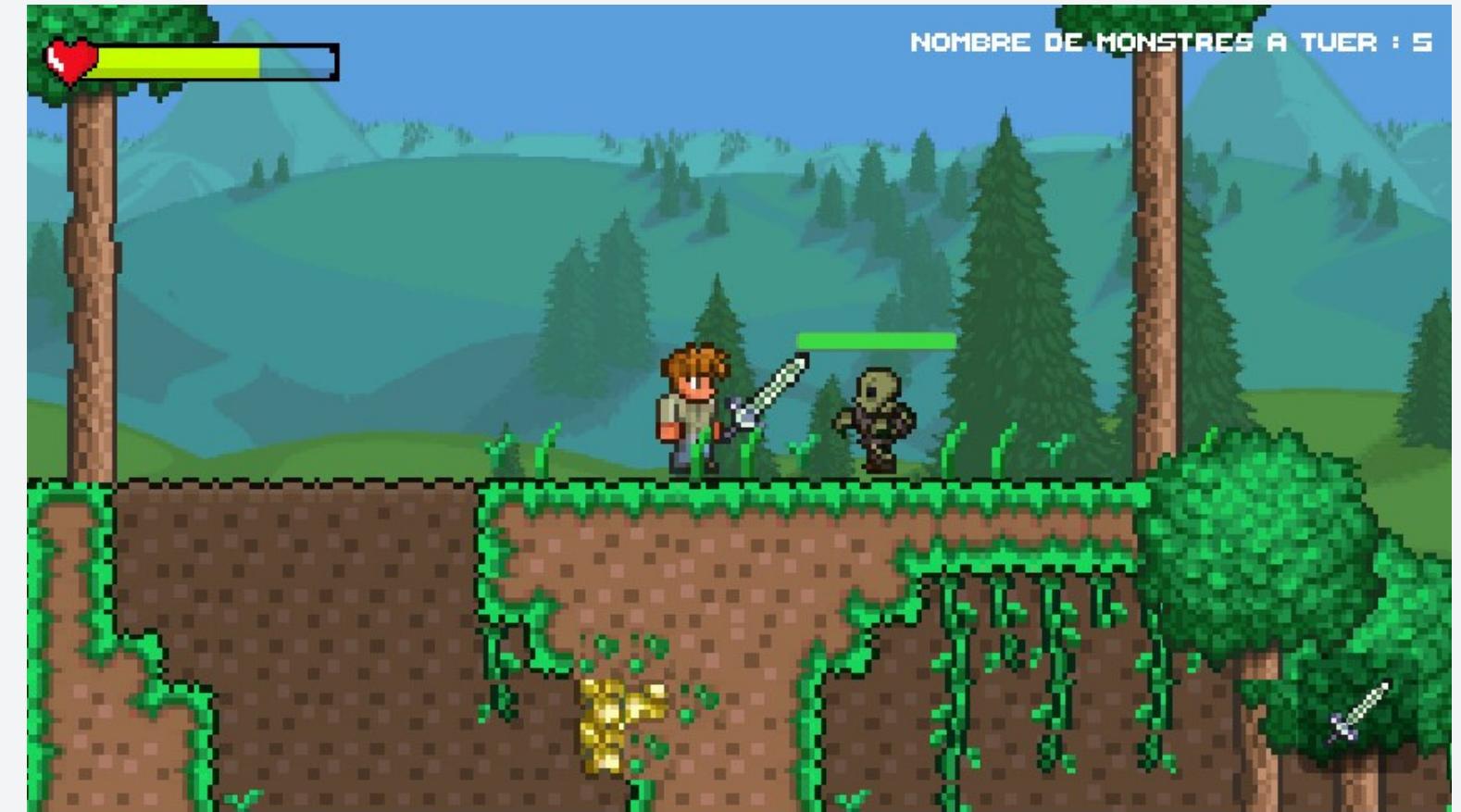
## 4.2 - Fonctionnalités

### ➤ Fonctionnalités du joueur

- Déplacements horizontaux
- Sauts
- Barre de vie
- S'équiper d'une arme (choix entre une épée et un pistolet) + possibilité d'utiliser une pioche

### ➤ Environnement

- Map générée de manière procédurale
- Monstres ayant une barre de vie, et pouvant infliger des dégâts au joueur si contact
- Les monstres poursuivent le joueur s'il est assez proche d'eux
- Compteur affichant le nombre de monstres à tuer pour passer au niveau suivant



# 5 - Conclusion et apprentissages

## ➤ Apprentissage de l'environnement Unity et C#

Système de boucle

Déclaration de variables publiques et privées

Bibliothèques C#

## ➤ Voies d'amélioration possibles

Coffres de loot disposés sur la map pour redonner de la vie au joueur

Nouveaux niveaux disponibles, avec une augmentation de la difficulté

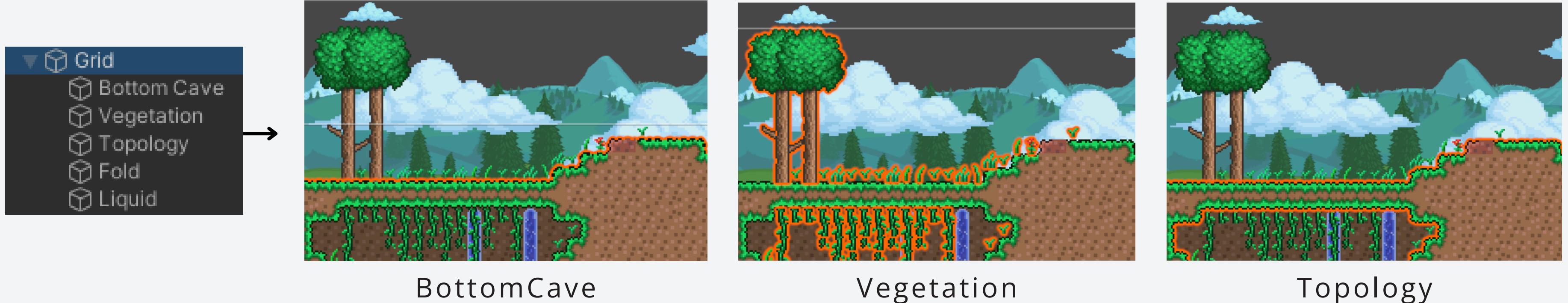
Différents skins du personnage à choisir

## 6 - LOTEREO : Le jeu

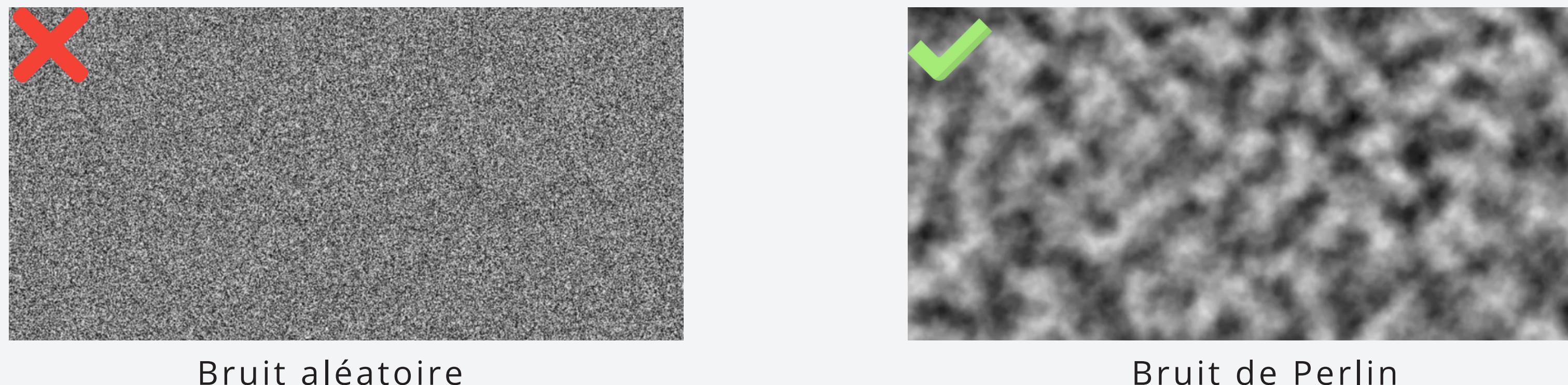


# 6.1 - Terrain : Différentes Tile map

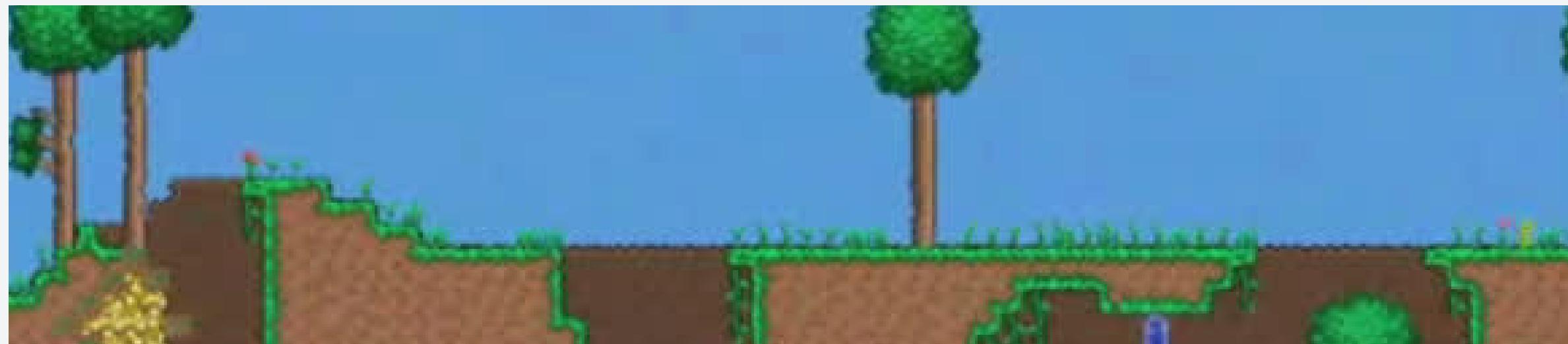
## ► Ordonnancement



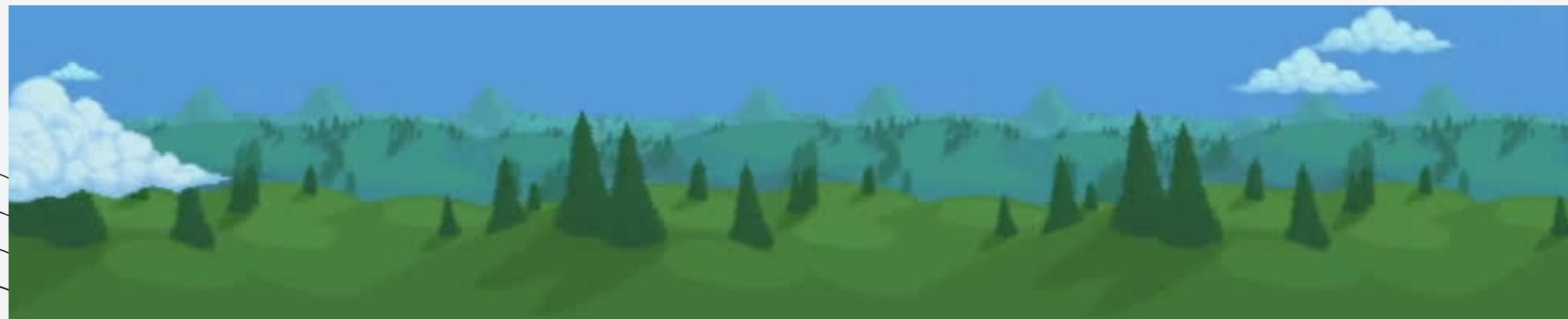
## ► Carte de Perlin



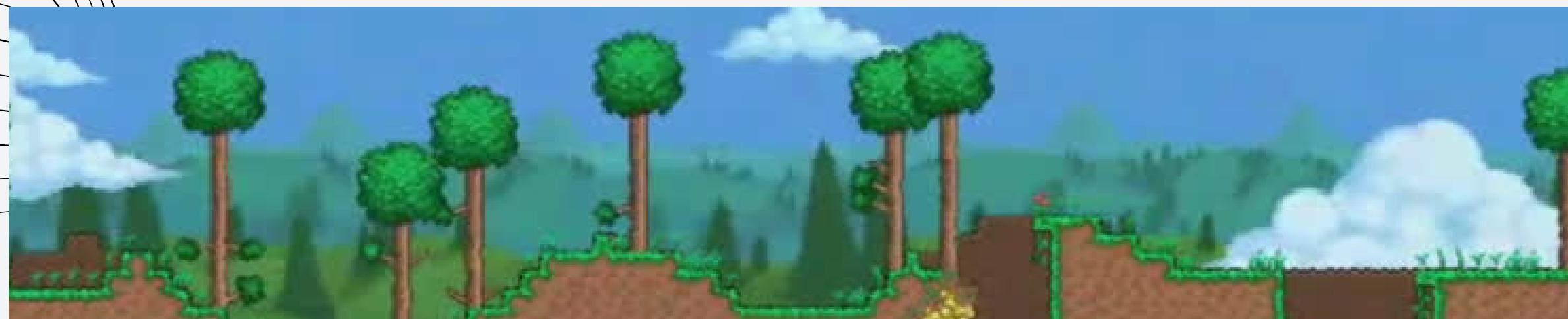
## 6.2 - Terrain : Rendu final



➤ Carte topologique

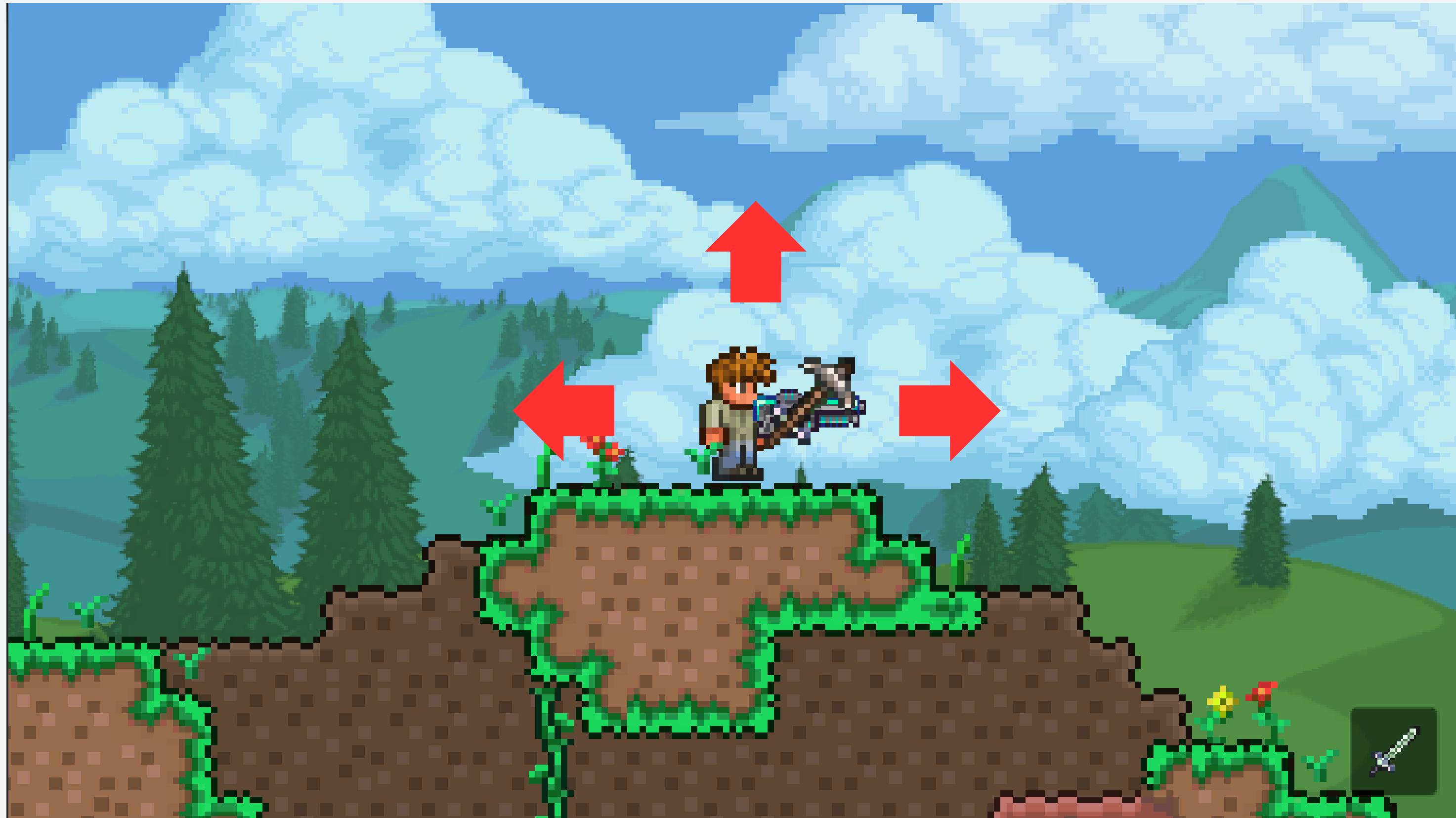


➤ Fond à effet parallaxe



➤ **Rendu final**  
Carte topologique +  
Fond à effet parallaxe

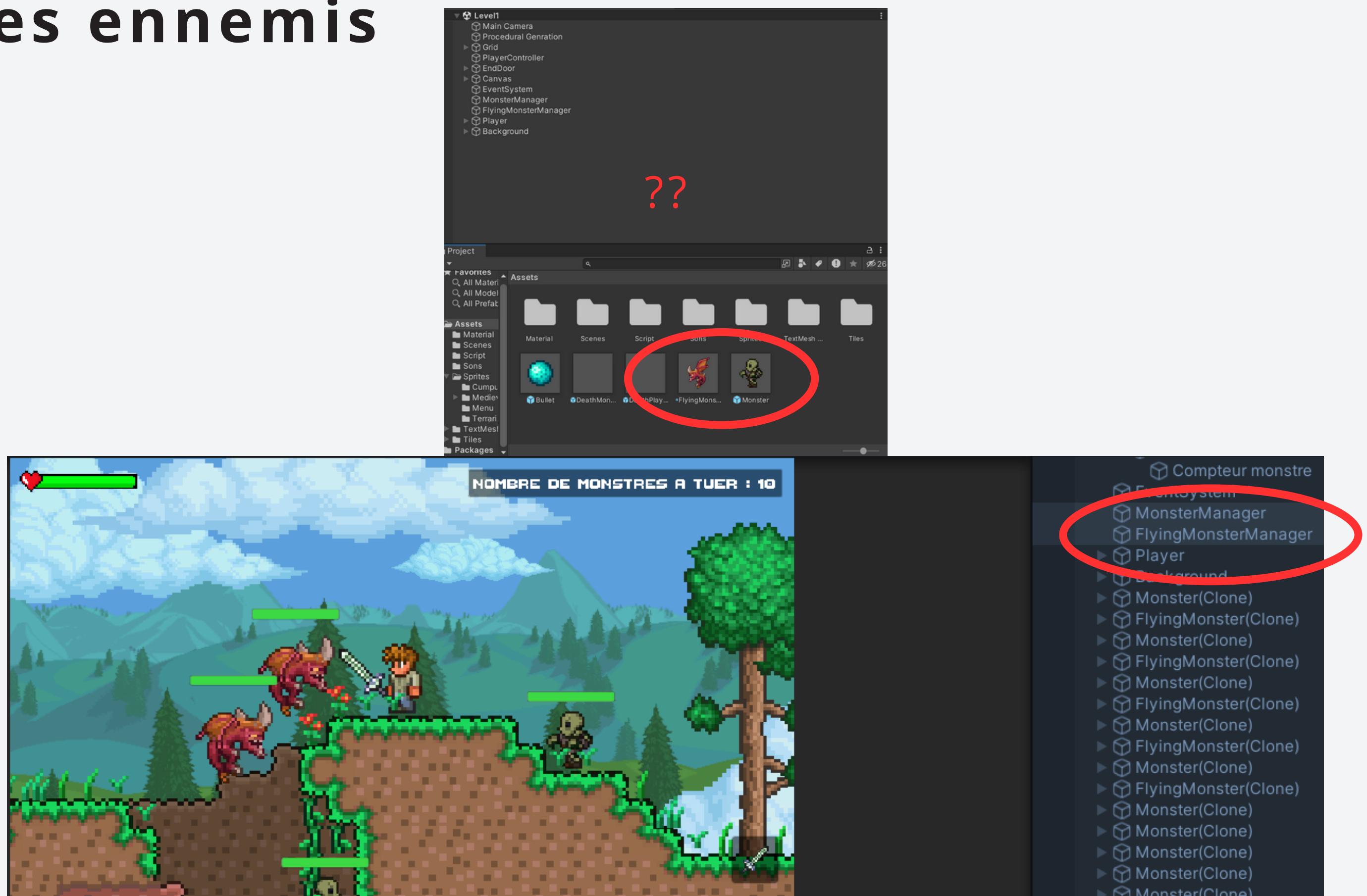
## 6.3 - Le Joueur



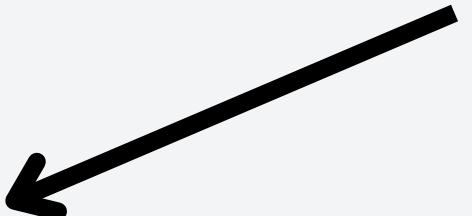
## 6.3 - Le Joueur



## 6.4 - Les ennemis



## 6.5 - L'objectif et les menus



Jouons !