NavMesh (PathFinding)

Pour créer des chemins de déplacement pour les personnages intelligents, il faut créer un mesh virtuel navigable (à l'aide de la fenêtre Navigation) et ajouter un composant *NavMeshAgent* sur le personnage IA.

Rappel des étapes **(pour tous les détails, voir le tutoriel vidéo du cours)**

Configuration et création du NavMesh

1. Dans la fenêtre de l'inspecteur, activez la propriété *Static* des objets statiques de votre environnement (typiquement le plancher, les murs et les différents obstacles qui ne bougent pas).
2. Ouvrez la fenêtre *Navigation* (Menu Window/AI/Navigation). Placez le à côté de l'Inspector.
3. Dans l'onglet *Object* de la fenêtre *Navigation*, déterminez pour chaque objet statique s'il est possible de marcher dessus ou pas.
   Sélectionner l'objet dans le *Hierachy* et modifier les propriétés zones (*areas*) pour : walkable, not walkable, etc.
4. Dans l'onglet *Bake* de la fenêtre *navigation*, ajustez les propriétés (*max slope, step height,* etc.)
5. Générez le *navMesh* en appuyant sur le bouton *Bake*.
6. Visualisez le *navMesh* créé et ajustez les différentes valeurs si nécessaire. **Dès qu'un changement est apporté ou qu'un nouvel objet est ajouté, il faut recréer le *navMesh* en appuyant sur *Bake*.**
   (si le NavMesh n'apparaît pas alors il faut activer son Gizmos dans la fenêtre Scene).

Configuration de l'agent *navMesh (le gameObject de AI)*

1. Dans la fenêtre de l'inspecteur, ajoutez le composant *NavMeshAgent* sur l'objet AI (*GameObject)* qui doit être contrôlé par l'agent *navMesh*.
2. Ajustez les différentes propriétés du composant *NavMeshAgent* (Radius, Speed, etc.)
3. Créez le script pour gérer le *NavMeshAgent*.

   Pour voir toutes les priorités et les méthodes de NavMeshAgent :
   https://docs.unity3d.com/ScriptReference/AI.NavMeshAgent.html

Donnez une destination à l'agent *navMesh (exemple)*

```
// Ce script déplace un objet vers un autre objet en utilisant le “path finding” de Unity i.e. NavMesh
// Le NavMesh doit être créé et un composant NavMeshAgent doit être ajouté à l’objet

using UnityEngine.AI;     //il faut ajouter le type AI de Unity
public GameObject MaDestination ;
NavMeshAgent navAgent ;

void Start ()
{
   navAgent = GetComponent<NavMeshAgent>();
}
void Update ()
{
     navAgent.SetDestination(MaDestination.transform.position); //la destination doit être un Vector3
     print(navAgent.velocity.magnitude);  //imprime la vitesse de déplacement de l’agent
     navAgent.enabled = false;                //désactive le fonctionnement de NavMeshAgent (le composant)
}
```

Consignes pour l'exercice (suite du projet "*SurvieShooter*")

Création du navMesh

Créez le *NavMesh*, en intégrant les objets de l'environnement qui doivent être *static*. Seulement l’objet *Plancher* doit permettre de marcher dessus (*Walkable*), le reste des objets sont *non-Walkable*.

Configuration de l'agent *navMesh*

- Placez les trois types d’ennemis à partir de leurs modèles de base sur la scène. (dossier Model/Character)
- Ajoutez et configurez le composant *NavMeshAgent* aux trois ennemis.
    - Ajustez le rayon, la hauteur et les autres paramètres des *NavMeshAgent*. L’Hellephant doit se déplacer et se tourner plus lentement que le ZomBear et le ZomBear doit se déplacer et se tourner plus lentement que le ZomBunny.
    - Créez l’ **Animator** de chaque Ennemi qui doit avoir 3 animations (Idle, Marche, Mort). l’animation par défaut est la “Marche”.
- Créez un script "ennemi" et associez-le aux trois ennemis. De cette façon, tous les ennemis générés posséderont une instance de ce script. **Le même script doit être utilisé pour les trois ennemis.**
- Dans ce script, écrivez le code nécessaire pour que l'ennemi se déplace à l'aide de l'agent *NavMesh* et ait comme destination le joueur principal. **Cette destination doit être mise à jour à chaque fram puisque le joueur se déplace.**
- Ajoutez une fonction *Touche()*. Nous la compléterons plus tard.
- Testez le bon fonctionnement des ennemis.
    - Ajustez le rayon, la hauteur et les autres paramètres des NavMeshAgent si nécessaire. L’Hellephant doit se déplacer et se tourner plus lentement que le ZomBear et le ZomBear doit se déplacer et se tourner plus lentement que le ZomBunny.
    - Désactiver les ennemis, car nous allons les générer dynamiquement.

Génération des ennemis

Générez des ennemis en suivant les directives suivantes :

- Créez un *gameObject* vide "GenerateurEnnemis".
- Créez un nouveau script "GenererEnnemisScript"et associez-le au *gameObject* "GenerateurEnnemis".
- Dans ce script, faites en sorte qu'un nouvel ennemi apparaît toutes les 2, ou 5 secondes. Les monstres les plus faibles (*ZomBear et ZomBunny*) doivent être générés à chaque 2 sec, et les plus résistants (*Hellephant) à 5 sec*. Astuce: Il faut utiliser InvokeRepeating() , qui doit appeler une fonctione qui instancie l'ennemi.

Mort des ennemis

Lorsque le personnage tire, il est maintenant nécessaire de savoir s'il a touché un ennemi.

- Commencez par modifier le code présent dans le script de la balle pour pouvoir détecter un ennemi.

  Astuce : vous pouvez attribuer un *tag* aux ennemis...

- Si un ennemi est touché par une balle :
  - Appelez la fonction *Touche()* de l'ennemi. L'objet touché est obtenu par infoCollision.gameObject de la fonctione de OnCollisionEnter() de la balle;

> **Rappel : Pour appeler une fonction dans le script d'un autre *gameObject***
>
> *referenceAuGameObject*.GetComponent<*NomDuScript*>().*NomDeFonction()*

- Lorsque la fonction *Touche()* d'un ennemi est appelée, dans cette fonction :
  - Le clip sonore (son de mort) qui est déjà associé au *gameObject* de l'ennemi doit se faire entendre.
  - L'animation de mort de l'ennemi doit s'activer.
  - L'ennemi doit arrêter de se déplacer vers le personnage principal (désactivation de l'agent à l'aide de NavMesh.enabled =false).
  - Faites en sorte que l'ennemi ne soit plus atteint par les tirs pendant son animation de mort (pour qu'une balle ne puisse pas détecter un ennemi. **Astuce :** modifier le Tag pour "Untagged").
  - Après un délai de 2 secondes, l'ennemi doit être détruit.

Défi optionnel - Différents niveaux de vie pour les ennemis

Lorsqu'un ennemi est touché :

- Diminuez la vie de l'ennemi touché.
  - *Hellephant* = 5 tirs pour mourir
  - *ZomBunny* = 2 tirs pour mourir
  - *ZomBear* = 1 tir pour mourir

- Si l'ennemi est toujours vivant, le son "d'ennemi touché" doit se faire entendre. Chaque ennemi possède son propre son d'ennemi touché qui est bien identifié dans l'onglet projet.
- Si l'ennemi doit mourir, les instructions créées précédemment doivent s'exécuter.
- Créez le système de particules (*DeathParticles)* qui doit s'activer. *DeathParticles* est un enfant de l'ennemi. Pour trouver et activer l'enfant d'un *gameObject*, vous pouvez utiliser la fonction : ***transform.Find("nomEnfant") ou transform.GetChild(NumeroEnfant) 0,1,2 etc***

```
var particule = transform.Find("DeathParticles");
```

Remarque: La particule est active, mais PlayOnAwake est décochée, alors on l'active comme un clip de son.