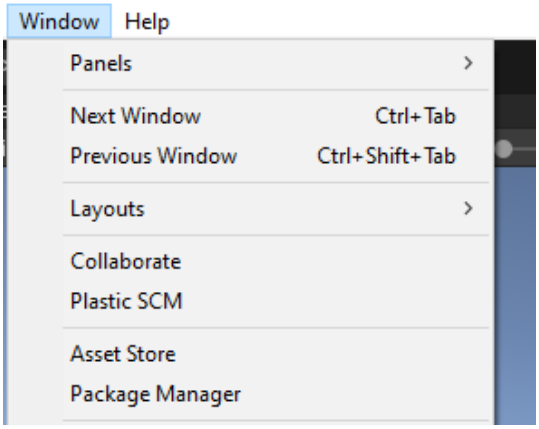
Jeux et simulations

Éric Wenaas
420-4GP-BB
Module 5 – Asset Store et animations
Version 2.0

Assets Store de Unity

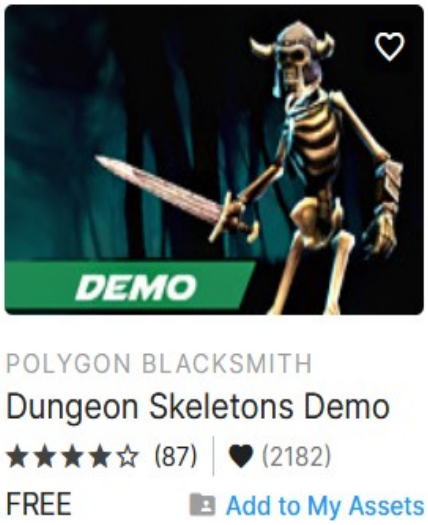
Unity offre une banque de modèles 3D et d'animations qui peuvent être utilisés dans les projets. Si plusieurs de ces assets sont payants, plusieurs sont gratuits.

On peut accéder à l'Asset Store directement à partir de Unity dans le menu Window:



L'asset Store s'ouvrira ensuite dans un navigateur. Faites une recherche de *Fantasy Mushroom Monster*.

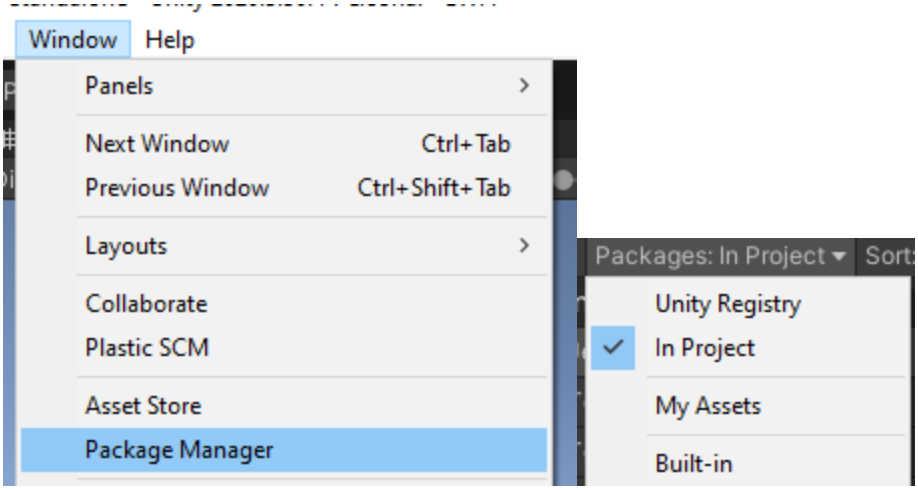
Faites l'acquisition de cet asset:



Afin d'utiliser un asset, il faut premièrement l'acheter (même si c'est gratuit)[1]. Ensuite, il faut le télécharger et l'importer dans notre projet.

Téléchargement de l'asset

Pour télécharger l'asset, il faut aller dans le package manager de Unity et sélectionner My Assets:



Retrouvez votre asset dans la liste et dans le coin droit vous aurez un bouton Download. Après le téléchargement, vous pourrez l'importer.

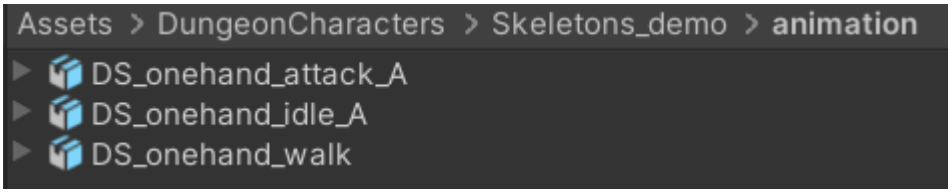
1 Pour les travaux, vous n'avez pas à débourser d'argent pour des assets. Vous n'aurez aucun point de plus si vous le faites.

Vous pourrez choisir ce que vous voulez importer dans votre projet. Ici, vous devez tout importe. Il existe des assets qui viennent également avec des scripts ou encore des exemple de scènes que vous n'importerez pas nécessairement toujours dans vos projets.

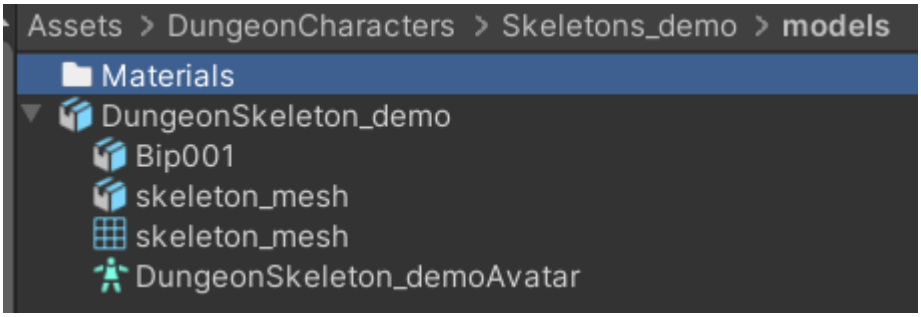
Une fois importée, le paquetage de l'asset sera dans un dossier. Vous pouvez réorganiser les assets si vous le désirez.

Le paquetage importé contient :

Des animations :



Un modèle et des matériaux :



Vous pouvez réorganiser ces assets dans votre stucture d'assets si vous le désirez.

Animations et le composant Animator

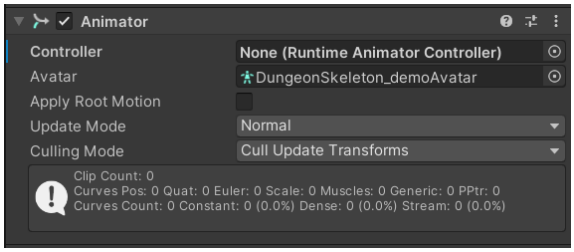
Le champignon comprend différentes animations:

- Attack
- Damage
- Death

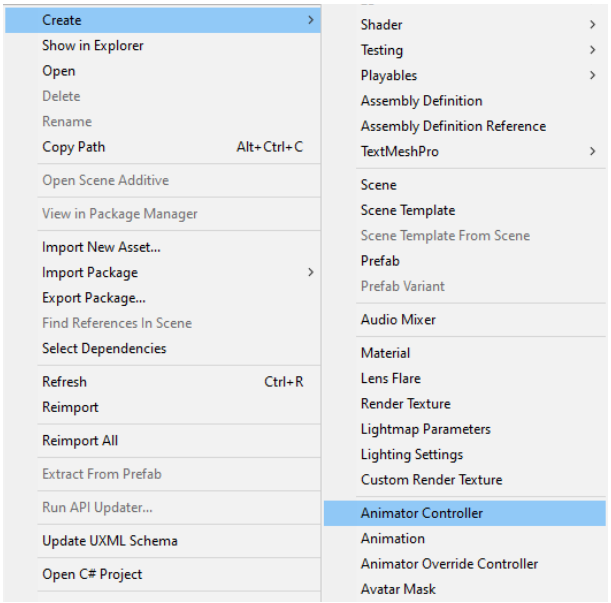
- Idle
- Run

Le composant Animator permet de construire un diagramme d'état qui permettra à notre modèle 3D de changer d'état quand certaines conditions se produisent.

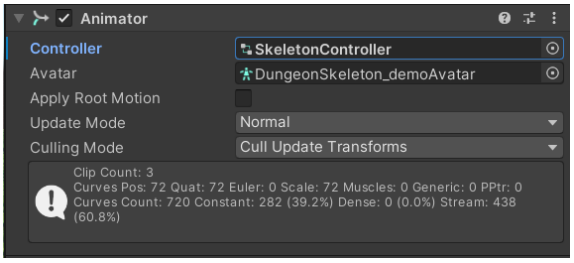
Un composant Animator doit être ajouté à un objet qui contient des animations. On doit ajouter un contrôleur pour spécifier les changements d'états:



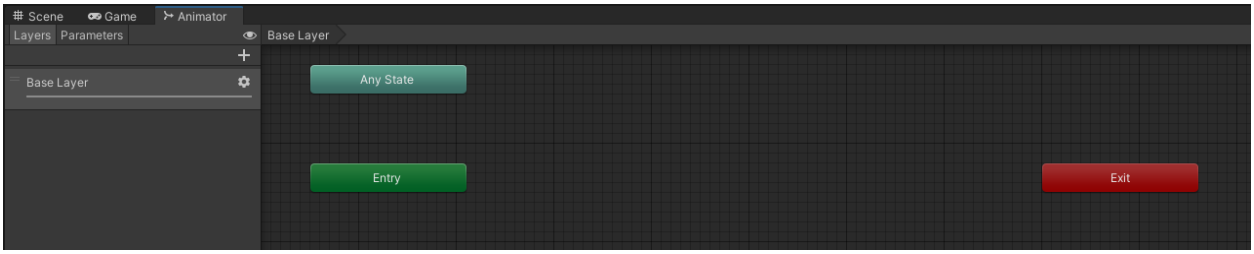
On doit créer un Animator:



Ce contrôleur devra ensuite être ajouté à l'objet:

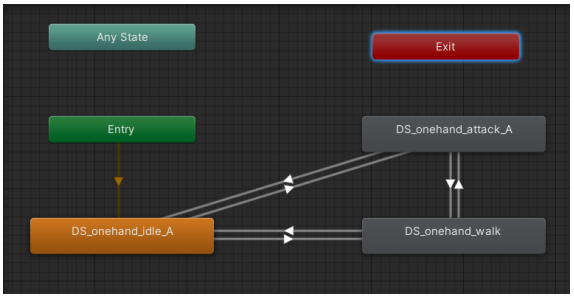


Un contrôleur est un diagramme d'état:



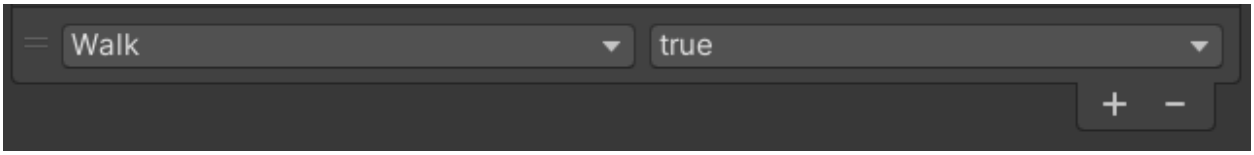
Un diagramme d'état contient des états, des transitions et des conditions pour effectuer une transition. Quand une transition se produit, le modèle changera d'animation. En cliquant à droite, on peut ajouter un nouvel état. Ensuite, on peut faire une transition entre deux états.

Par exemple:



Ceci signifie que l'état d'entrée est Idle. De l'état Idle, on peut aller à Walk ou Attack. Un changement d'état va se produire par une transition qui elle dépend d'un paramètre. Pour contrôler la transition, nous allons créer une valeur booléenne se nommant Walk:

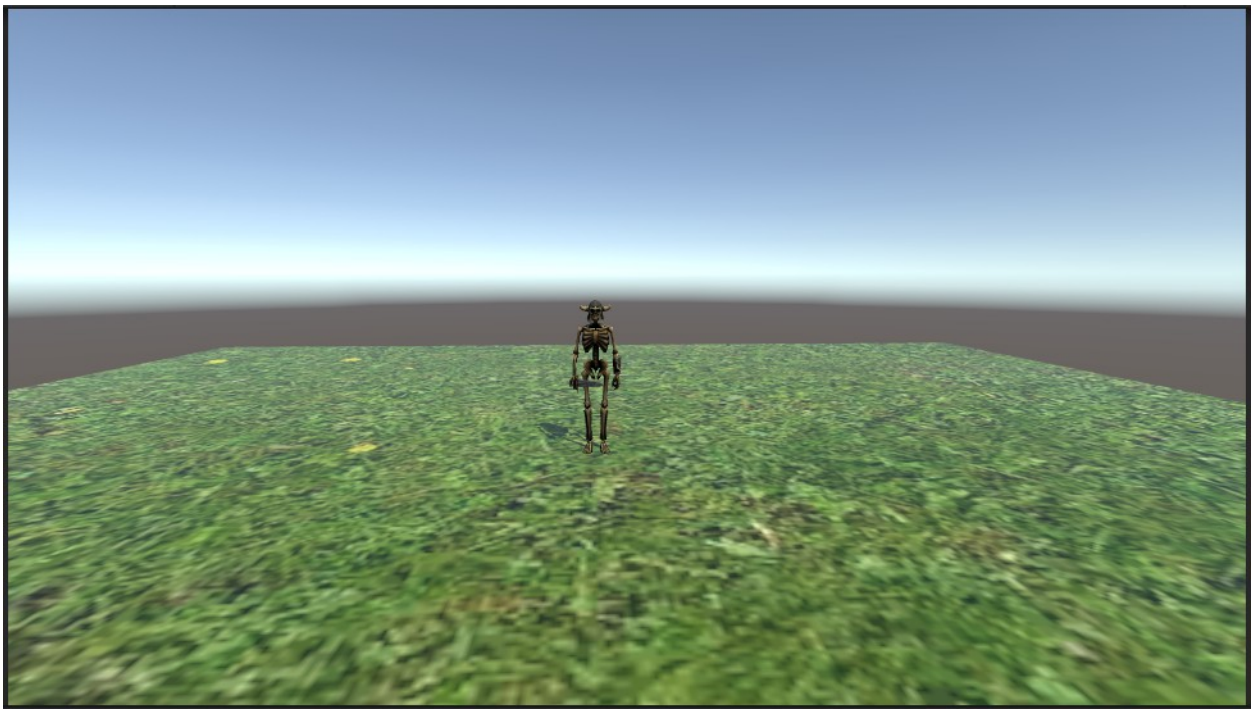
Ensuite, on sélectionne la transition entre deux états et on spécifie la condition qui doit être vraie pour que la transition s'effectue:



Ici, le contrôleur d'animation passera automatiquement à l'animation spécifiée dans l'état Walk lorsque la valeur booléenne de l'animateur deviendra true. Pour le faire revenir dans l'état Idle, il faudra faire le contraire dans la transition de Walk à Idle.

Exercice #1

Réalisez cette scène:



Pour mettre une texture sur le plan, recherchez une texture de gazon, importez-la avec Assets→Import New Assets… Ensuite, vous glissez l'image sur le plan.

Vous devez animez le squelette de la façon suivante:

- Par défaut, si l'utilisateur ne fait rien, l'animation est Idle
- Si on fait la touche A, l'animation est Attack. Après l'attaque, on revient dans le même état où il était.
- Si le squelette se déplace, l'animation est Walk. Faites en sorte que le squelette se déplace quand l'utilisateur clique sur le plan.

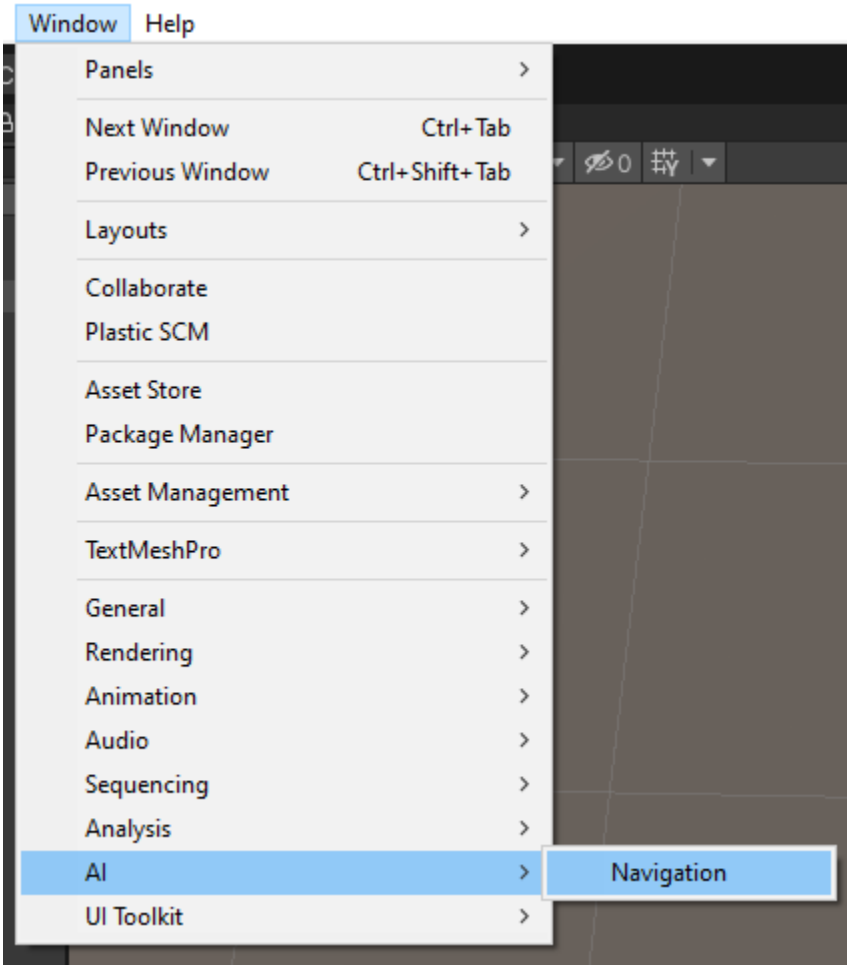
Faites le contrôleur d'animation et le code pour réaliser ces spécifications.

NavMesh Navigation

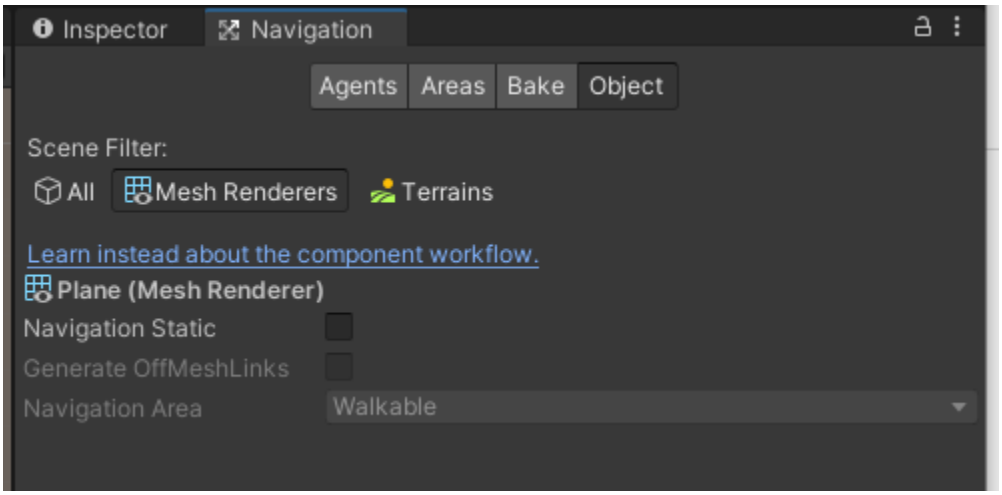
La classe NavMeshAgent est un comportement qui peut être ajouté à un objet de jeu. Ce comportement permet à un objet de se déplacer seul en contournant les obstacles. Il s'agit en fait d'une application automatique d'algorithmes de Pathfinding.

Pour activer le Pathfinding sur un objet, il faut suivre certaines étapes:

Premièrement, il faudra ouvrir la fenêtre de Navigation dans le menu Window:



Ensuite, il faudra indiquer au moteur de Pathfinding quels seront les objets qui doivent servir à générer la zone où on peut circuler . Pour ce faire, on utilise le paramètre Navigation Static qu'on retrouve dans l'onglet Navigation:

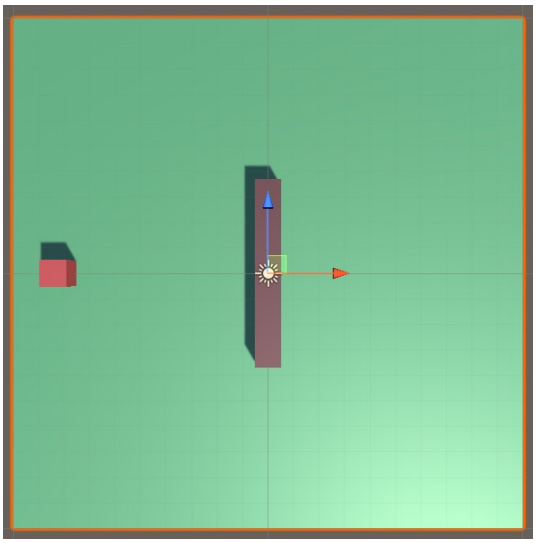


Une fois, que les éléments ont été correctement sélectionnés. On doit faire l'opération Bake. Ce qui nous donne une zone dans laquelle un objet peut se déplacer.

Pour qu'un GameObject utilise le NavMesh, on doit ajouter à l'objet le composant NavMeshAgent.

Exercice #2

Commencer par réaliser cette scène:

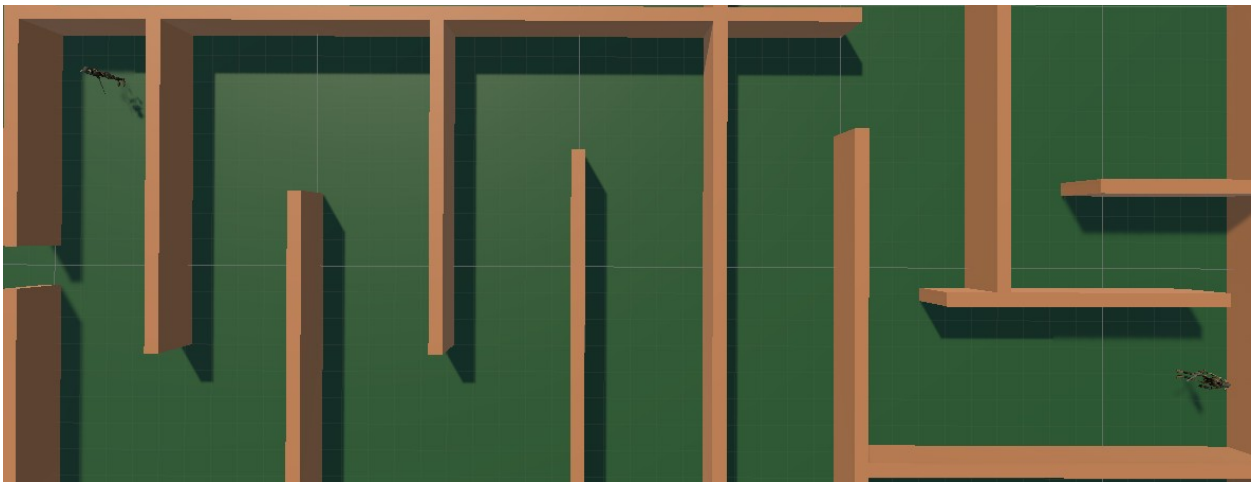


Vous devez faire en sorte que lorsqu'on clique sur le plan, le cube se déplace automatiquement sur le point cliqué en contournant l'obstacle s'il y a lieu. Vous devez utiliser les NavMesh à cette fin.

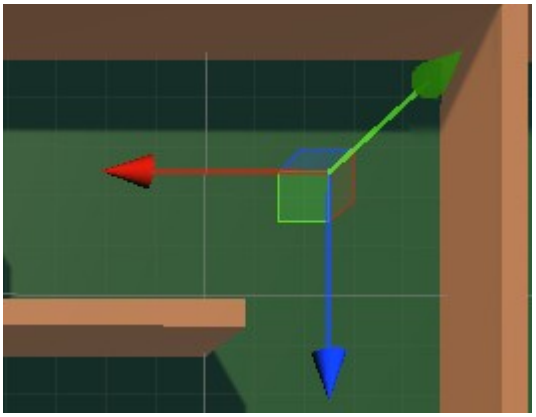
Exercice #3

Commencez par récupérer les packages Depart5_3.unitypackage et Squelette.unitypackage fournis avec le module. Nous allons maintenant introduire les squelettes dans le labyrinthe. Ils patrouilleront le labyrinthe afin de capturer le joueur.

Par exemple, vous pouvez ajouter les squelettes suivants:



Les squelettes devront faire une patrouille dans le labyrinthe afin d'attraper le joueur. Un point de patrouille sera simplement un objet vide sur le labyrinthe :



Les points de patrouille seront conservés un tableau. Ajoutez un comportement dans une classe Patrouille.cs aux squelettes. Lorsqu'un squelette est rendu au dernier point de patrouille, il revient tout simplement au point de départ. Le déplacement se fera avec un NavMesh.

Faites également en sorte que si le joueur entre en contact avec les champignons, il perd la partie. S'il touche à l'objectif, il gagne la partie. Dans les deux cas, il faudra afficher une scène de victoire ou de défaite. La scène est affichée pendant 5 secondes et ensuite on revient simplement à la scène du menu. Vous pouvez utiliser les images qui sont fournies avec ce

module.

Vous pouvez également ajoutez les textures pour le plancher et les murs.

Patron de conception État (State Pattern)

Le patron État est très utile dans les jeux. Ce patron est particulièrement bien adapté aux situations où un objet doit changer de comportement selon certaines situations. Par exemple, dans le contexte d'un jeu, un personnage peut devenir plus fort s'il utilise un certain objet, il peut devenir plus faible s'il est blessé, etc.

Dans ce patron, des classes sont définies afin de décrire l'état dans lequel un objet de jeu se retrouve. Ces classes héritent toutes de la même classe mère. Ceci permet d'appliquer un comportement à un objet sans avoir à connaître l'état dans lequel on est.

La force de ce patron est que l'objet pourra changer d'état de façon transparente. De cette façon, en utilisant le polymorphisme, on pourra appeler une méthode sur un objet sans connaître la nature de l'instance qui sert d'état.

Le diagramme de classe de ce patron se retrouve ici. Le contexte est l'objet qui possède un état . Par exemple, un joueur. La référence sur cet état est du type de la classe abstraite.

Exercice #5

Modifiez le comportement des squelettes. Il y aura trois comportements possibles

Patrouille

Le squelette fait sa patrouille normalement. Ceci est l'état par défaut. L'animation doit être Walk.

Poursuite

Le squelette se dirige vers le point où il a vu le joueur la dernière fois. En mode poursuite, le squelette suit donc le joueur. Pour voir le joueur, il doit être dans un angle de 60 degrés de son vecteur forward. L'animation doit être Walk.

Attente

Si le champignon était en mode Poursuite et qu'il ne voit plus le joueur, il s'arrête et attends de 3 à 5 secondes. Ensuite, il retourne faire sa patrouille. L'animation doit être Idle.

Vous devez réaliser cet exercice en appliquant le patron état.

Attaque

Si le squelette voit le joueur et qu'il est à moins de 2.5 m de distance, il attaque dans la direction du joueu