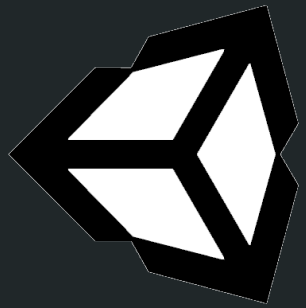
Introduction à Unity

Les NavMeshAgents



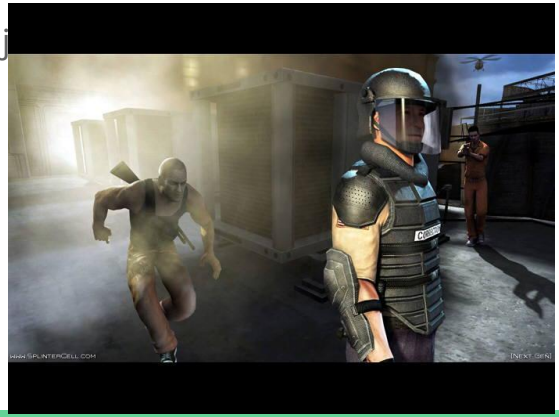
Un NavMeshAgent ?

- C’est un outil Unity qui permet de faire du pathFinding pour le déplacement d’entité.
  - Du pathQuoi ? Le pathfinding est la technique utilisé pour trouver le chemin le plus rapide entre deux points sur une map avec (ou sans) obstacle.

    L’idée est de trouver des points de repère sur une grille pour faire des points de passage.

    Ensuite en explorant les distances et les poids de chaque point on cherche le plus court chemin (optionnel : voir théorie des graphes (math) ou algorithme A* et dij

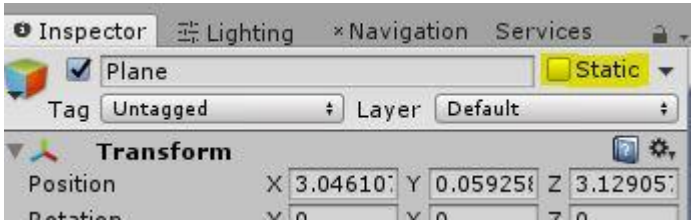
- Le navMeshAgent est un composant qui se met sur un GameObject
  - Add Component ➜ Navigation ➜ NavMeshAgent



Le navMeshAgent

- **Agent size :** définit les différentes données de l'agent à changer en fonction de l'objet
- **Steering :** définit les options de déplacement de l'agent vitesse, vitesse pour tourner, accélération, distance d'arrêt
- **Obstacle avoidance :** quelle méthode va être utilisé pour calculer les possibles collisions pour la qualité est haute plus c'est coûteux en temps de calcule
- **Path Finding :** les options de calcule de la trajectoir (off mesh link & area mask expliqué ci dessous) et recalcule automatique

Carte de navigation



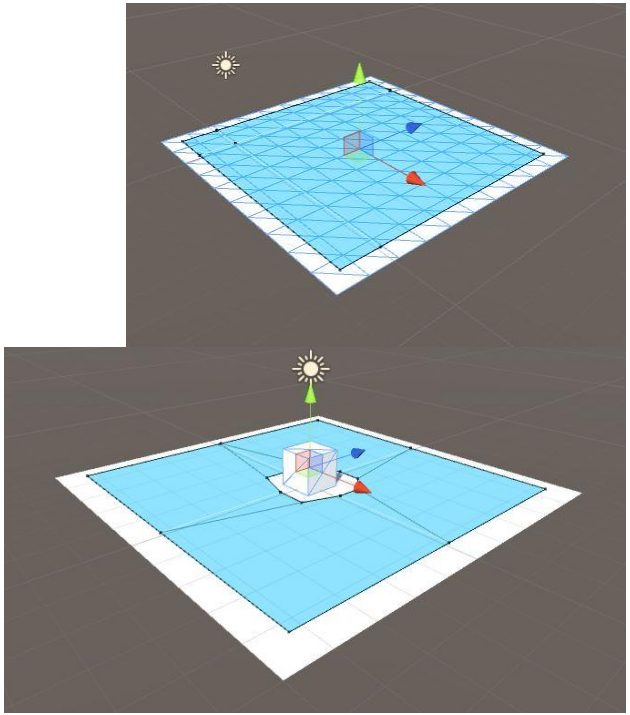
- Comme dit ci-dessus les navMesh ont besoin d'avoir un espace découpé en point de passage pour naviguer.

- Posons un plan
  - Vous devez obligatoirement cocher la case static en haut à droite du gameObject
  - Static veut dire que votre objet ne bougera pas pendant la durée du jeu, il est très important pour l'optimisation du moteur donc les murs / décors etc doivent être mis en static.
  - Dans l'onglet Navigation à gauche de l'inspector (cf. capture ci-dessus ou cours 1 ajouter des fenêtres)
  - Clicquer Bake en bas à droite

Ajouter des obstacles

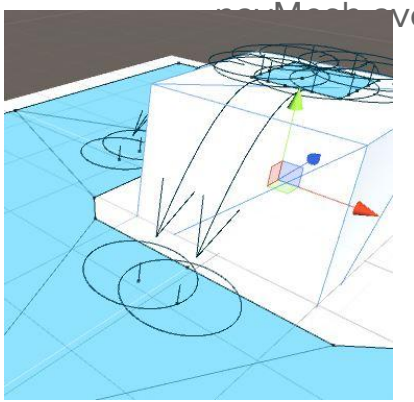
- Ajouter un cube sur le plan
- Mettez le static
- Refaire le bake

- Le résultat est semblable à la figure 2
- Vous voyez apparaître les lignes reliant les points de passage qui découpe la surface de déplacement

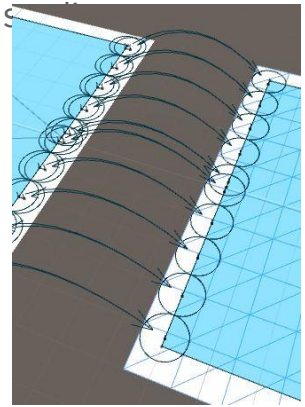


Plus en détail

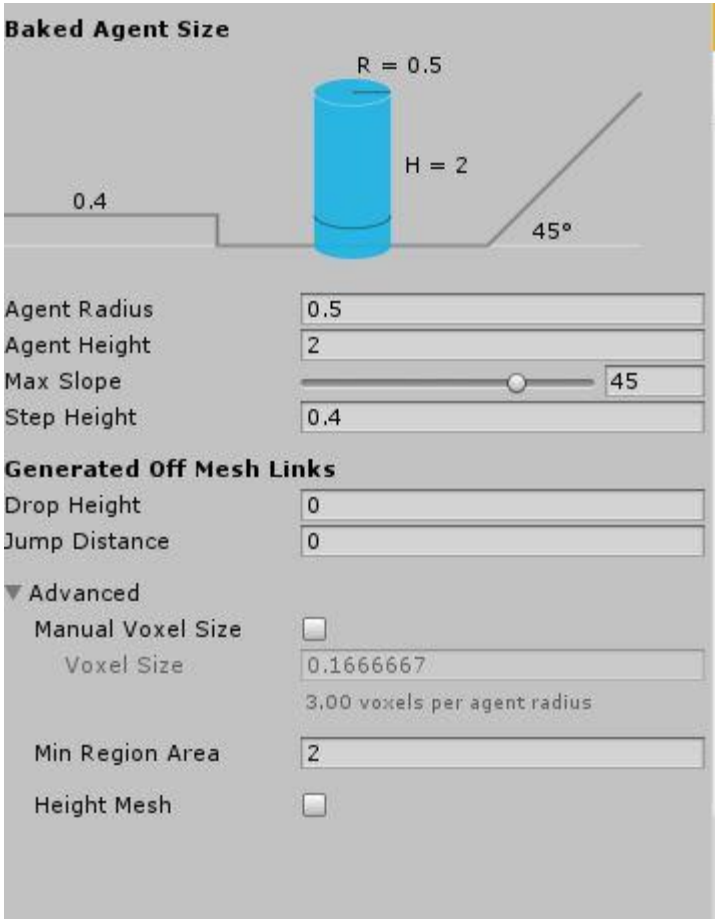
- offMeshLinks
  - Ce sont des connections entre les différentes zone du navMesh créer avec la fonction Bake
  - Drop Height définit la capacité à connecter un navMesh avec un autre par rapport à sa hauteur
  - JumpHeight définit la capacité à connecter un navMesh avec un autre par rapport à s



← Drop Height



Jump Height →



Optionnel : Voxel

- Dans la fenêtre de Bake - Advanced vous pouvez voir Manuel Voxel size

- Le voxel est une technique utilisé notamment dans Minecraft pour faire des petites zone amovible

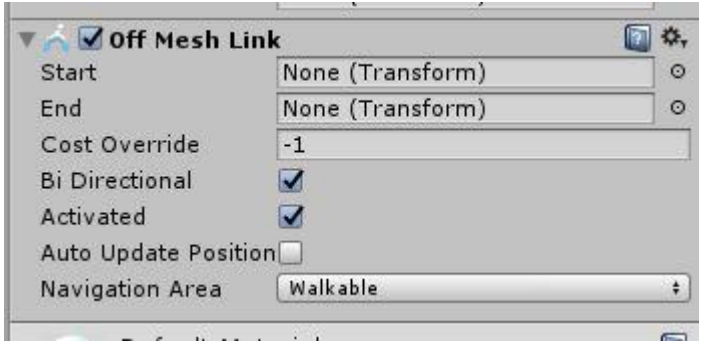
- L’idée est ici de mettre des cubes côte à côte pour créer un terrain, ce qui pour un navMesh classique poserait des problèmes pour créer la carte de navigation

- En modifiant ces options vous pouvez créer un terrain Minecraft-like

Manuel Off-Mesh Link

- Unity peut créer des off mesh link automatiquement
- Mais parfois ce n’est pas suffisant alors il existe le composant off Mesh Link qui permet de définir des points de saut
- Il suffit de lui passer un transform de départ et d’arriver
- Si l’objet ne contient pas de visuel on utilisera un 3D Object ➜ Empty pour ne pas gêner le joueur

- Attention il faut que le transform soit proche du sol Bake pour fonctionner

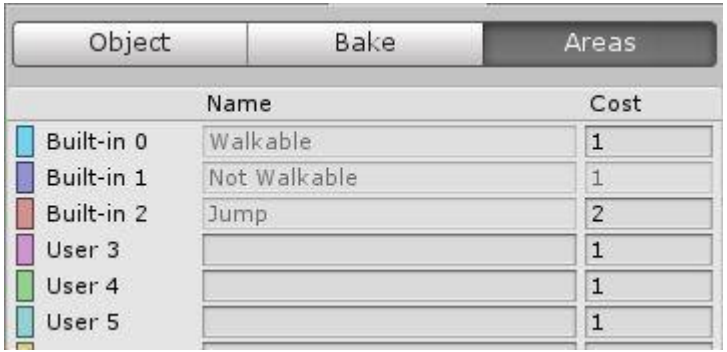


Area Cost

- Dans les off Mesh Link Manuel vous avez du remarquer la case Cost Override

- Dans un jeu, si les unités sont dans la boue, dans un escalier ou autre, le coup de déplacement n’est pas forcément le même.

- Vous pouvez définir des coups par zone et les affecter dans l’onglet Object
  Attention cependant à bien avoir selectionné l’objet désiré avant de changer le coup



Limitation et solution des NavMesh

- Le bake ne peut se faire que dans l’editor Unity, ce qui veut dire que vous ne pouvez pas avoir de monde créé aléatoirement ?

- Il existe les navMeshObstacle qui viendront modifier en temps réel le navMesh déjà Bake.

- Le navMeshObstacle est un composant à mettre sur un objet pour qu’il soit pris en compte dans le calcule des trajectoires.

Côté programmation

- Les navMeshAgent sont utilisé pour les PNJ qui doivent se déplacer.

- Ainsi, ils vont être amené à changer de destination souvent il va falloir gérer ça dans le code via la fonction :
  - SetDestination(Vector3 destination)
  - Le navMesh va se charger de recalculer tout seul son chemin
  - Le navMesh peut être passé en argument SerializeField (ou public selon le besoin)
  - La fonction Stop pour arrêter la navigation vers la destination (garde la destination en mémoire)
  - La fonction Resume pour recommencer à bouger vers la destination

Maintenant : A vous de courir !