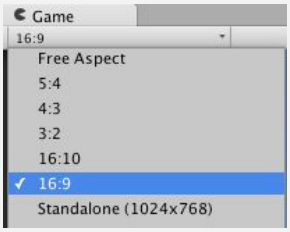
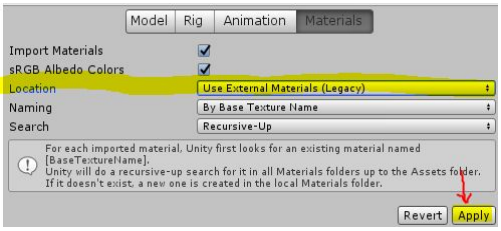
Exercice Hélico, partie I

Avant de commencer, il est toujours recommandé de fixer la taille de l'onglet game à une certaine valeur et de ne pas travailler en "free aspect". Pour cet exercice, choisissez le ratio 16:9



Modifier le point de pivot de l'hélice arrière dans Maya

- Pour simplifier le script de rotation des hélices, il faut avoir le même axe de pivot pour les 2 hélices,
- Vous pouvez les modifier dans Maya et l'exporter en format .fbx .
    - Ouvrez un projet vide Maya
    - Importer l'helico.fbx
    - Sélectionnez l'hélice arrière
    - Affichez un manipulateur (exemple : Touche W)
    - Touche D pour faire apparaître le pivot
    - Orientez le pivot pour avoir Y comme axe de rotation,
    - Menu/Modify/Bake pivot
    - Enregistrez le projet.
    - File/Export All , en format fbx (ou lieu de .mb)
    - Importez l'hélico.fbx dans Unity
        - ajustez le scale factor,
        - dans l'onglet **Materiels** modifiez **Location**:
          (sinon le matériel ne peut pas être modifié)



Créer une plate forme d'atterrissage

- Pour permettre à l'hélico de rester stable lors de l'atterrissage, créez un objet cylindre dans unity , modifiez sa taille, son collider pour un Mesh Collider (supprimer le Capsule Collider qu'il possède), et sa texture (choisissez n'importe quelle texture disponible dans le Project).

Faire tourner les hélices de l'hélico

- Créez un script C# : **onglet Project/Create/C# Script** et nommez-le : « TourneObjet »
- Ouvrez le fichier en double-cliquant dessus,
- Déclarez une variable publique ***vitesseRotation*** de type **Vector3**. (Un Vector3 contient les 3 valeurs x, y et z) (on peut le faire avec un float mais on veut pratiquer les Vector3).

- Dans la fonction ***Update()***, ajoutez les instructions pour:
  - Faire tourner l'hélice selon la valeur de la variable déclarée précédemment en utilisant **transform.Rotate().**
  - Enregistrez et attachez (glissez) le ce script sur l'hélice avant et l'hélice arrière dans l'onglet *Hierarchy*. Ces éléments font partie de l'objet *Helico*.
  - Ajustez les valeurs de la variable vitesseRotation dans l'inspecteur
    Attention si les axes de rotation des deux hélices ne sont pas les mêmes alors
    vous avez besoin du nom de l'objet pour savoir selon quelle axe il faut tourner l'hélice :
    if(gameObject.name == "HéliceHaut") .....
  - En appuyant sur le bouton *Play* les deux hélices doivent tourner.

Ajouter un effet d'accélération

- Pour plus de réalisme, les hélices ne devraient pas tourner dès le départ. Modifiez le script pour que la variable ***vitesseRotation*** de l'hélice augmente progressivement de 0 jusqu'à une valeur maximale. Cette valeur maximale pourra être définie dans une variable (***vitesseRotationMax type float).***

Démarrage et arrêt du moteur

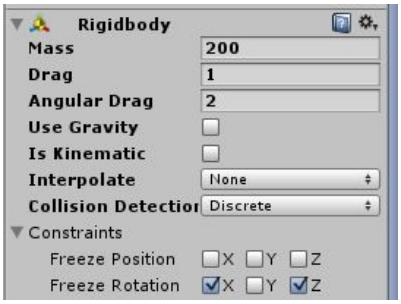
- Modifiez le script pour que la touche "*Return*" permette de démarrer ou d'arrêter les l'hélices :
  - Détecter la touche "*Return*", en utilisant ***Input.GetKeyDown()***
  - Lorsque cette touche est appuyée, l'objet (l'hélice) doit commencer à tourner... Si l'hélice tourne déjà, elle doit alors s'arrêter progressivement...
    Indice:
    Il faut utiliser une variable *demarreMoteur* pour mémoriser l'état de rotation (true ou false)!
    À l'aide de ***print***(*demarreMoteur*) vérifiez dans le console si la valeur change avec chaque input.

Partie II : Gestion de déplacement de l'hélico

Les composants nécessaires: Rigidbody, Mesh Collider, ScriptDeplacement

Les instructions nécessaires: FIxedUpdate(), Input.GetAxis(), Input.GetKey(), AddRelatvieForce(), AddRelaticeTorque(),

1. Ajoutez un composant Rigidbody sur l'hélico (**masse** = 200, **Drag** = 1, **AngularDrag** = 2, **Gravity** = false, **FreezRotation** sur X et Z)



2. Ajoutez un MeshCollider sur l'hélico **!! l'option *Convex* doit être toujours cochée pour un Rigidbody !!**

3. Créez un script C# *DeplacementHelico*;
4. Assignez ce script à l'hélico, (en le glissant sur l'objet *Helico* dans l'onglet *Hierarchy*).
5. Déclarez quatre variables de type *float* pour contrôler les vitesses de l'hélico :
   vitesseAvant = 0f; vitesseAvantMax = 10000f; vitesseTourne = 1000f; vitesseMonte = 1000f;
   Les valeurs de ces variables dépendent de la masse et des différentes frictions (*drag et angularDrag*) de l'objet. Essayez différentes valeurs pour voir les résultats.

6. Toutes les instructions qui manipulent le déplacement à l'aide des forces doivent se trouver dans **FixedUpdate**().

7. La vitesse pour tourner l'hélico selon l'axe Y est déterminée à l'aide des touches "a" et "d" et les flèches gauche et droite, en utilisant l'instruction : **Input.GetAxis("Horizontal")** ;

8. Cette vitesse (force) de rotation est appliquée en utilisant l'instruction **AddRelativeTorque()** ;
   Modifiez le paramètre **AngularDrag** dans l'inspecteur pour trouver la rotation qui vous plaît.

   ------- la suite peut être fait au prochain cours: Avancer, Monter/Descendre ----------

9. Une seule instruction **AddRelativeForce()** permet d'appliquer les vitesses pour avancer et pour monter/descendre l'hélico. Déterminez toutes les valeurs ensuite appliquez la force.
   Remarque 1: il ne faut jamais avoir plus d'une instruction AddRelativeForce() (ou AddForce() dans un script.
   Remarque 2: il ne faut jamais déplacer un rigidbody en modifiant sa transform.position ou par Translate. Sinon ça peut créer des bugs de collisions.

10. La vitesse pour monter et descendre l'hélico est déterminée à l'aide des touches "w" et "s" (et les flèches haut et bas), en utilisant l'instruction **Input.GetAxis("Vertical")**,

    Cette vitesse est influencée par la vitesse vers l'avant. Si la vitesse avant est grande alors l'hélico monte

(ou descend) plus rapidement. (Cette partie est à ajouter quand la vitesse avant fonctionne)
(indice: il faut ajouter une fraction de la vitesse avant à la vitesse monter)

11. La vitesse pour faire avancer l'hélico est déterminée à l'aide des touches "q" et "e" :
    - lorsque la touche "e" est appuyée, cette vitesse augmente jusqu'à la vitesse maximale
    - lorsque la touche "q" est appuyée, cette vitesse diminue jusqu'à 0.

12. **Démarrage et arrêt du moteur (suite)**

    Le **useGravity** doit être coché dans l'inspecteur au début du jeu.
    Si l'hélice avant tourne à sa vitesse maximale alors l'hélico peut avancer et monter/descendre sinon l'hélico tombe, en mettant la propriété **useGravity** de son Rigidbody à *true*. Sinon si la vitesse diminue alors la gravité s'active.
    Pour accéder à la valeur d'une variable dans un autre objet, il faut utiliser :

    ReferenceAutreGameObject.**GetComponent<***NomDuScript***>().***nomDelaVariable*

    Exemple

```
public GameObject RefHelice ; // Permet de glisser dans l'inspecteur un autre gameObject. On aura
                              // ainsi la référence à ce dernier dans  le script courant.

// Pour récupérer la valeur de la variable vitesseDeRotation du script TourneObjet qui est sur le
// gameObject de l'hélice avant ou arrière. Les instructions qui suivent doivent être placées dans le
// FixedUpdate() de l'hélico.
var vitesseHelice = RefHelice.GetComponent<????> ().?????.?
...
if(vitesseHelice > ??)
//  Si la condition est vraie, il faut alors désactiver la propriété useGravity (false). À vous de trouver
l'instruction!
//Sinon…
```

13. Positionnez la caméra derrière l'hélico et définissez-la comme enfant de l'hélico, en la glissant sur l'objet hélico.
14. Correction de Bug de Constraint de Unity:
    Les conrtaints sur la rotation en X et Z ne fonctionnent pas toujours bien. Alors, pour empêcher l'hélico de se pencher, on fixe l'angle en Y à la rotation actuelle et on met les axes X et Z à 0 avec :
    transform.localEulerAngles = new Vector3(0 , transform.localEulerAngles.y ,0);

    transform.localEulerAngels est la même chose que **transform.rotation** mais en degré.
15. Enregistrez la scène et sauvegarder votre projet sur vos lecteurs.