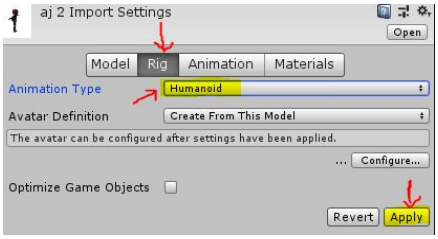
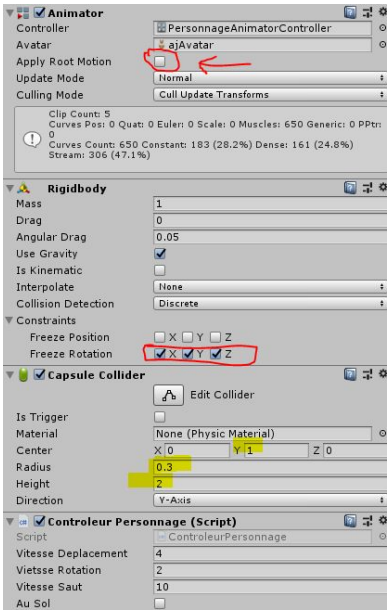
Déplacement d'un personnage

Les composants nécessaires pour configurer un personnage:



- Un personnage (en .fbx) configuré dans l'inspecteur en humanoid (ou Generic),
- un Rigidbody,
- un Capsule Collider,
- un contrôleur d'animation (onglet Project/Create/Animator Controller),
- un script de contrôle de déplacement et d'animation.



Les types de déplacements:

Le choix dépend du style de jeu (*GamePlay*) désiré. On peut déplacer une personnage par l'**ajout de force** ou en **modifiant la vélocité** du personnage.

1- Ajout de force: (Hélico, vaisseau, auto etc.)

Personnage qui va réagir avec les objets physiques de façon réaliste (ex. se fait frapper par une roche qui roule, doit pousser des objets, qui a une accélération ou un ralentissement graduels, qui dérape, etc.).

GetComponent<RigidBody>().AddRelativeForce() ou GetComponent<RigidBody>().AddForce()

Remarque: Pour avoir un déplacement très rapide et un arrêt rapide du personnage (sans glissement), il faut augmenter le Linear Drag de beaucoup (exemple 10), et ajuster les autres propriétés physiques.

Exemples:

```
public Rigidbody rigidbodyPerso;       // contiendra la référence au composant rigidbody de l'objet
float vitesseDeplacement = 10f;       //multiplicateur de vitesse de déplacement

void Start()
{
   rigidbodyPerso = GetComponent<Rigidbody>();      //on assigne à la variable le composant Rigidbody d'un objet 3D
}
```

```
// Applique une force sur  l'objet selon les touches verticales (w,s, flèche haut et bas),  dans l'axe Z continuellement
void FixedUpdate()
{
   var vDeplacement = Input.GetAxis("Vertical") * vitesseDeplacement;
   rigidbodyPerso.AddRelativeForce(0, 0, vDeplacement );      //applique une force dans l'axe de Z  de l'objet

  // ou
   rigidbodyPerso.AddForce(0, 0, vDeplacement );      //applique une force dans l'axe de Z  du monde

  // ou
   rigidbodyPerso.AddForce(vDeplacement, 0, 0 );      //applique une force dans l'axe de X  du monde

}
```

2- Modifier la vélocité: (Déplacement d'un personnage Humanoïde)

Personnage qui se déplace aussitôt que les touches de déplacement sont appuyées. (Pour les jeux à actions rapides)
Il faut modifier sa vélocité physique: **GetComponent<RigidBody>().vélocité = Valeur sous forme de Vector3**

Contrairement au Add**Relative**Force, il n'y a pas de commande qui permet de modifier la vélocité selon les axes locaux d'un objet. Si on désire faire avancer un personnage selon son axe Z, il est nécessaire d'utiliser le vecteur **transform.forward** pour obtenir la direction de l'axe Z du personnage.

Remarque: Le code peut être dans Update().
Les interactions physiques ne sont pas aussi précises qu'avec l'ajout de force.

```
public Rigidbody rigidbodyPerso;       // contiendra la référence au composant rigidbody de l'objet
public  float vitesseDeplacement = 100f;       //multiplicateur de vitesse de déplacement

 void Start()
 {
    rigidbodyPerso= GetComponent<Rigidbody>();      //on assigne à la variable le composant Rigidbody d'un objet 3D
 }
 // Déplace l'objet selon les touches verticales,  avance dans l'axe Z continuellement
 void Update()
 {
    var vDeplacement = Input.GetAxis("Vertical") * vitesseDeplacement;

   //fixe la vélocité dans l'axe de Z  de l'objet   , qui fournit par son transform.forward
    rigidbodyPerso.velocity= transform.forward * vDeplacement;
   //ça remplace:   rigidbodyPerso.AddRelativeForce(0, 0, vDeplacement )

   // OU si le personnage doit subir la gravité et pouvoir sauter alors il faut ajouter sa vélocité en y
    rigidbodyPerso.velocity = (transform.forward * vDeplacement) + new Vector3(0, rigidbodyPerso.velocity.y, 0);
```

```
    if (Input.GetKeyDown(KeyCode.Space))
    {
       rigidbodyPerso.velocity += new Vector3(0, 10 ,0);
    }
  //OU s'il faut déplacer le personnage selon les axes du monde
    rigidbodyPerso.velocity = new Vector3(0,0, vDeplacement);//applique une force dans l'axe de Z  du monde
    //remplace :  rigidbodyPerso.AddForce(0, 0, vDeplacement );
}
```

3 - Rotation du personnage sans force de torsion (torque est très lent):

A. Personnage qui tourne autour de lui même:

Dans la majorité des cas, lorsque la rotation est déterminée à l'aide des touches (exemple Input.GetAxis("Horizontal") ou à l'aide de la souris (Input.GetAxis("Mouse X"), on peut utiliser:

La valeur de **tourne** est déterminée par:
var tourne = Input.GetAxis("Horizontal") * vitesseRotation;
ou
var tourne = Input.GetAxis("Mouse X") * vitesseRotation;

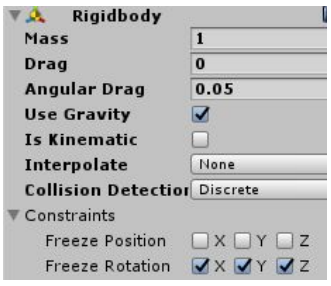
Remarque: Input.GetAxis("Mouse X") renvoie une valeur négative ou positive selon le déplacement gauche ou droit de la souris. Il existe aussi le "Mouse Y".

Applique la rotation:
transform.Rotate(0, tourne , 0) ; // ici tourne est une variable de type float

B. Personnage qui s'oriente vers une position (ex: regarder vers la position de la souris, ou un objet) ou une direction (exemple: selon la direction de la caméra) alors on peut utiliser :

transform.LookAt(unePosition); // une position est un Vector3
ou
transform.rotation = Quaternion.LookRotation(unePosition);
ou
transform.forward = maCamera.transform.forward //regarde vers le devant de la caméra (son axe Z)

Remarque : La propriété *Freeze Rotation* de l'objet doit être cochée pour X, Y et Z, car ce n'est plus l'engin physique qui gère les rotations



Gestion des animations d'un personnage

Configuration du contrôleur des animations (Animator Controller)

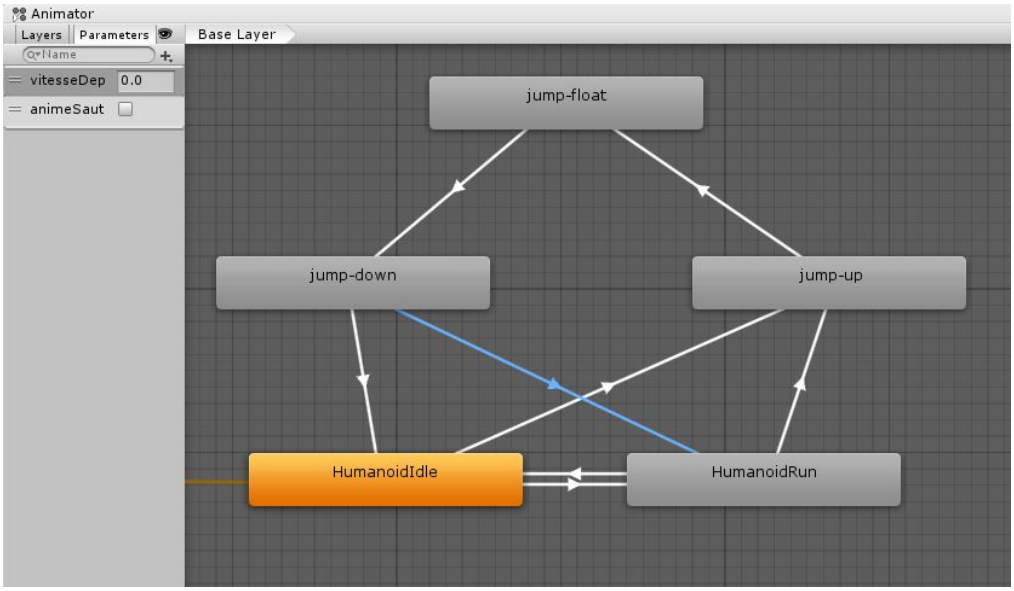
Voici un exemple de configuration d'un *animator* qui peut s'appliquer à un personnage en général.

Le personnage possède les animations: **Repos** (Idle), **Course** (Run), **Saut-haut** (jump-up), **En-Vol** (Jump-float) et **Atterrissage** (Jump-Down).

- le paramètre *vitesseDep* (float) représente la vitesse de déplacement et détermine si l'animation de *Repos* ou de *Course* doit jouer. Il peut aussi être utilisé pour la marche.

- Le paramètre *animSaut* (booléenne) détermine si la séquence d'animations de saut doit jouer.

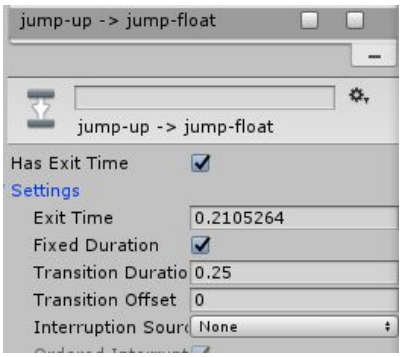
- Le saut peut commencer à partir de l'animation de *Repos* ou de l'animation de *Course*. Si le personnage ne touche pas le sol alors ce paramètre *animSaut* devient true. Et lorsqu'il touche le sol alors il devient false.



Configuration des transitions:

Has Exit Time :

Si cette case est cochée alors l'animation doit jouer un certain temps (selon Exit Time) avant de passer à la prochaine animation.
Si elle n'est pas cochée, il faut une condition pour passer à la prochaine animation.



Transition Duration :

Détermine la durée de transition entre les deux animations.
L'Animator crée des animations intermédiaires durant ce temps pour permettre un changement d'animation plus fluide.

Voici les valeurs des paramètres pour les transitions de l'animation du personnage.

Transition	Has Exit Time	Conditions	Settings
Idle- Run	false	vitesseDep > 0.1	les valeurs par défaut
Run-Idle	false	vitesseDep < 0.1	les valeurs par défaut
Idle - Jump_Up Run - Jump_Up	false	animSaut = true	Trans. Duration = 0.05
Jump-Up - Jump-float	true	aucune	les valeurs par défaut
Jump-float - Jump_Down	false	animSaut =false	Trans. Duration = 0.05
Jump_Down - Idle	false	vitesseDep < 0.1	les valeurs par défaut
Jump_Down - Run	false	vitesseDep > 0.1	les valeurs par défaut

Exemple de modification des paramètres d'un *Animator* à partir du code du personnage:

```
  // Déplace l'objet selon les touches verticales,  avance dans l'axe Z continuellement ….
 void Update()
 {     ……..
  //Gestion de l'animation de Repos et course
   animPerso.SetFloat("vitesseDep", vDeplacement);

  /*******************Gestion de l'animation de Saut.*/
  // Détermine si le personnage touche le sol. Lance un rayon vers le bas à l'aide de
 // SphereCast(position, rayon, direction, infoCollision, distance)

    RaycastHit infoCollision;            //            position,                        rayon, direction   , infoCollision,
distance
     bool auSol = Physics.SphereCast(transform.position + new Vector3(0,0.5f,0), 0.2f, -Vector3.up, out infoCollision, 0.8f);

  // Si  le personnage n'est pas au sol alors l'animation de saut doit jouer sinon elle arrête
    animPerso.SetBool("animSaut", !auSol);
```

Exercice:

- Téléchargez le projet Unity qui se trouve sur le site.
- Configurez le personnage, (Humanoid, Capsule Collider, **Rigidbody sans "Apply root motion**")
- Créez son contrôleur d'animation comme indiqué plus haut pour (idle, course et les sauts),
  Les animations se trouvent dans le dossier Animation de l'onglet Project.
- Créez le script de contrôle du personnage pour le déplacer en modifiant sa vélocité selon son axe de devant et de le tourner à l'aide de la souris **Input.GetAxis("Mouse X") et** des touches horizontale. (voir plus-haut)
- La caméra de suivi (3ePersonne) est déjà présente dans le hiérarchie , activez son script pour qu'elle suit le personnage,
- Enregistrez la scène et gardez votre projet.