***Inverse Kinematics***

**Vocabulaire :**

* **Links :** des objets rigides.
* **Joints :** lie les « links » entre-eux.
* **End effector :** c’est la cible d’un joint.
* **Chains :** la liste de corps rigide qui compose un système d’IK.
* **2 types de joints :**
* **Revolute :** permet d’effectuer des rotations.
* **Prismatic :** permet d’effectuer des translations.
* **Optimizing :**
* C’est le fait de chercher une fonction de minimization ou maximization.
* **Exemple dans le sytème d’IK :** nous cherchons à minimizer la distance entre le end effector et la cible.

**Méthodes pour refaire son sytème d’IK :**

* Réseau neuronal.
* Coordonnées cyclique.
* Pseudo inverse.
* Dampted Least Squares.
* Analytic Two-Bone 2D.
* Etc..

**Difficultés rencontrées :**

* Demande une bonne compréhension des *mathématiques*.
* Des fois les *algorithmes* présentées et trouvées sur des articles s’avérées *faux* et il fallait *réévaluer* le résultat de calculs et *d’équations*.

**Conclusion :**

* L’IK est un système mathématique qui *peut s’avérer lourd* selon les algorithmes *car* il doit bien *souvent* être *joué* *à* *chaque* *frame*.
* Ce système permet de faire interagir un corps rigide avec son environnement et permet donc de *faire* *gagner* *une* *application* *en* *réalisme*.
* Il *ne* faut *pas* *l’utiliser* sur des modèles *trop* présents dans une application afin de ne pas vite *saturer* les *performances* de votre programme.
* Faire un système *d’IK* *avancé* *réaliste* qui gère un maximum de cas s’avère une *tâche* *complexe*, selon la qualité du rendu que l’on souhaite prétendre il faut *soit* *payer* un *plugin* *soit* *recoder* *le* *nôtre*.
* Néanmoins plusieurs *algorithmes* *2D* *peuvent* *s’implémenter* assez *vite* grâce à la *multitude* *d’articles* qui *traitent* le *sujet* de façon découpé et itérative.
* Personnellement, j’ai trouvé ce *projet* très *intéressant* et m’a permis d’avoir une *idée* *plus* *globale* sur les *animations* et il m’a forcé à *me* *mettre* en *difficulté* en *lisant* et comprenant de la *documentation* *technique*.
* Dans mon cas j’ai tenté de *réaliser* *2* *algorithmes* sur de la 2D :
* *Cyclic* *Coordinate* *Descent*.
* *Analytic* *two*-*bone* IK.
* Il en existe des tas d’autres.

**Quelle est le problème que résout l’IK :**

* Trouver les angles et position de chaque joint composant notre chaine de joints pour placer correctement chaque cible des liens.

**Notes intéressantes :**

* Lorsque l’on résoud un système d’IK on part toujours du dernier joint qui va pointer vers une cible (exemple bras, avant bras, main, la main est le dernier joint -> on part de façon inverse que le FK.).
* Ce dernier joint ou effector (cible d’un lien) traque la position et la rotation de cette cible.
* Il est possible que la position de la cible soit hors de portée.

***Inverse Kinematics sur Unity***

**Process pour fournir des animations en inverse kinematics :**

* Dans l’animator du personnage : cocher IK Pass.
* Dans l’inspector > sélectionner l’animator > apply root motion.

**Méthodes d’IK :**

* Voir OnAnimatorIK.

**Regarder un objet :**

If (null != lookTransform)

{

Animator.SetLookAtWeight(1.0f) ;

Animator.SetLookAtPosition(lookTransform.position) ;

}

**Tenir un objet :**

Pour tenir un objet, il faut toujours que l’objet que l’on souhaite tenir est un transform en enfant qui sera un place holder de la position à tenir. (holdTransform).

If (null != holdTransform)

{

animator.SetIKPositionWeight(AvatarIKGoal.RightHand,1);

animator.SetIKRotationWeight(AvatarIKGoal.RightHand,1);

animator.SetIKPosition(AvatarIKGoal.RightHand,rightHandObj.position);

animator.SetIKRotation(AvatarIKGoal.RightHand,rightHandObj.rotation);

}

Parcontre, à chaque fois que l’on ne souhaite pas avoir de IK sur une partie du corps, il faut redéfinir l’IK à sa position initiale (weight 0).

Else

{

animator.SetIKPositionWeight(AvatarIKGoal.RightHand,0);

animator.SetIKRotationWeight(AvatarIKGoal.RightHand,0);

animator.SetLookAtWeight(0);

}

**Intégration du Foot IK :**

* Regarder mon package sur le Foot IK.
* **Pour réaliser ce dernier j’ai :**
  + Ouvrir la scène Animation Controller.
  + Rajouter sur mon terrain un layer Terrain.
  + **Intégrer un personnage que j’ai télécharger sur Mixamo :**
    - **Animation type :** humanoid.
    - **Avatar definition :** create from this model.
    - Copier tous les composant du personage “U\_Character\_REF ».
    - Dans le composant Animator > Avatar : sélectionner selui que vous venez de créer à partir de votre nouveau personnage.
    - Rajouter enfant des pieds de mon personnage des place holders : « Right Foot Ray et Left Foot Ray ».
    - Activer l’IK sur tous les layers de l’animator de mon personnage.
    - Lui rajouter le tag Player.
  + Rajouter le script Foot IK sur mon personnage.
    - **Cliquer sur composant :**
      * Activer iKActive.
      * Sélectionner le layer Terrain.
      * Drag & drop les place holder « Right Foot Ray et Left Foot Ray ».
    - Modifier FootLOffset et FootROffset en run-time de sorte que les pieds soient parfaitement positionné, si la rotation des pieds ne sont pas bonnes, il faudra modifier la rotation « Right Foot Ray et Left Foot Ray ».
    - Pour que le script marche, il faudra faire en sorte qu’il rentre au bon moment dans les méthodes OnIdleIK, OnWalkIK et OnRunIK du script FootIK.cs.