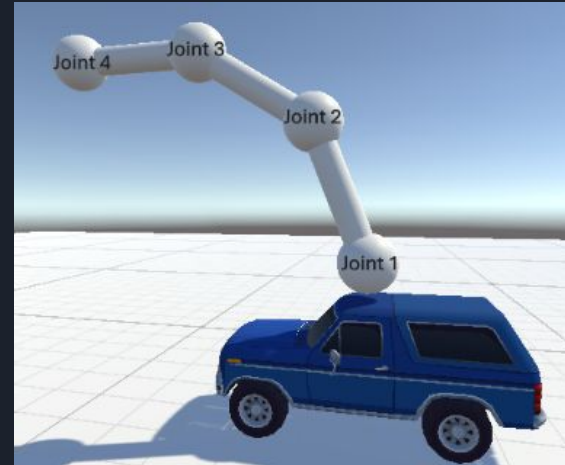




# **Simulation D'un Bras 3D Articulé Sous Unity**

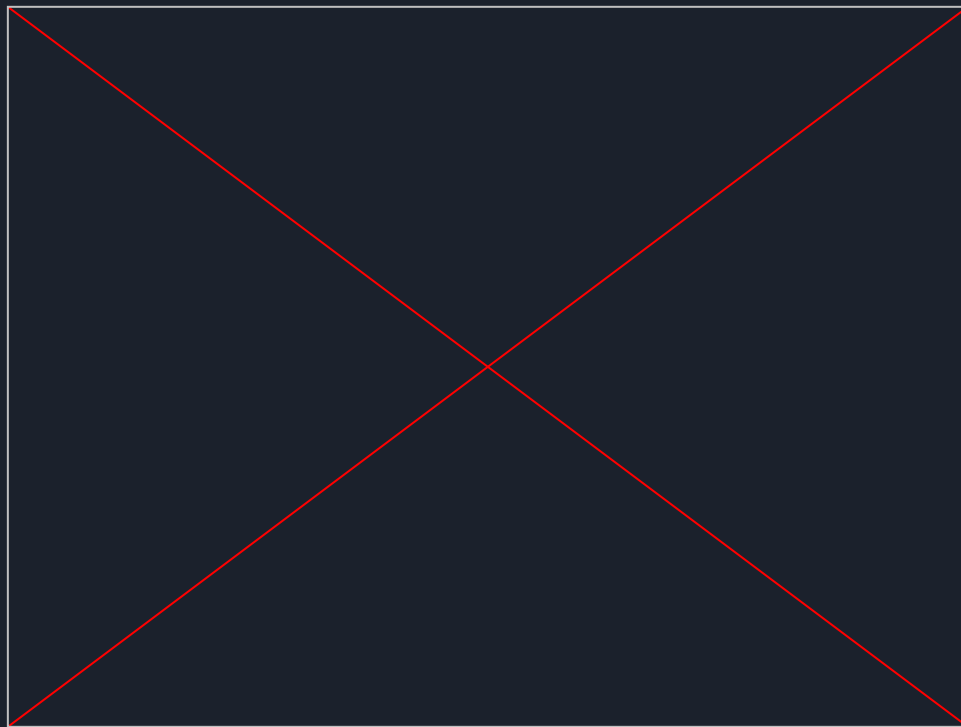
# Sommaire

- 1 - Démonstration
- 2 - Modèle direct
- 3 - Modèle inverse
- 4 - Trajectoire





# Démonstration



# Modèle direct

- Création de classes pour la matrice de translation et rotation

```
public struct RotationMatrixX
{
    3 references | 3 references | 3 references | 3 references
    public float m00, m01, m02, m03;
    3 references | 3 references | 3 references
    public float m10, m11, m12, m13;
    3 references | 3 references | 3 references | 3 references
    public float m20, m21, m22, m23;
    2 references | 2 references | 2 references | 2 references
    public float m30, m31, m32, m33;

    1 reference
    public RotationMatrixX(float angle)
    {
        float rad = Mathf.Deg2Rad * angle;
        float cos = Mathf.Cos(rad);
        float sin = Mathf.Sin(rad);

        m00 = 1; m01 = 0; m02 = 0; m03 = 0;
        m10 = 0; m11 = cos; m12 = -sin; m13 = 0;
        m20 = 0; m21 = sin; m22 = cos; m23 = 0;
        m30 = 0; m31 = 0; m32 = 0; m33 = 1;
    }
}
```

```
public struct TranslationMatrix
{
    3 references | 3 references | 3 references | 3 references
    public float m00, m01, m02, m03;
    3 references | 3 references | 3 references | 3 references
    public float m10, m11, m12, m13;
    3 references | 3 references | 3 references | 3 references
    public float m20, m21, m22, m23;
    2 references | 2 references | 2 references | 2 references
    public float m30, m31, m32, m33;

    1 reference
    public TranslationMatrix(Vector3 position)
    {
        m00 = 1; m01 = 0; m02 = 0; m03 = position.x;
        m10 = 0; m11 = 1; m12 = 0; m13 = position.y;
        m20 = 0; m21 = 0; m22 = 1; m23 = position.z;
        m30 = 0; m31 = 0; m32 = 0; m33 = 1;
    }
}
```

On effectue un enchaînement de rotation et de translation sous la forme :

$$\underbrace{{}^0T_1(q_1) \mathcal{R}(\vec{z}, q_1) \mathcal{T}(\vec{x}, L1) \mathcal{R}(\vec{z}, q_2) \mathcal{T}(\vec{x}, L2) \mathcal{R}(\vec{z}, q_3) \mathcal{T}(\vec{x}, L3)}_{{}^0T_E(q)}$$



# Modèle direct

- Fonction de translation

```
public Matrix4x4 computeTranslation(float distx, float disty, float distz, Matrix4x4 pos)
{
    TranslationMatrix trans = new TranslationMatrix(new Vector3(distx, disty, distz));
    return pos * trans.ToMatrix4x4();
}
```

- Fonction de rotation

```
public Matrix4x4 computeRotationPosition(float tetaX, float tetaY, float tetaZ, Matrix4x4 position)
{
    return position * new RotationMatrixX(tetaX).ToMatrix4x4() * new RotationMatrixY(tetaY).ToMatrix4x4() * new RotationMatrixZ(tetaZ).ToMatrix4x4();
}
```



# Modèle inverse

Nous avons opté pour un blocage du dernier joint sur un angle à  $180^\circ$  vers le bas, ce qui nous a permis de suivre les étapes suivantes :

1. Alignement du bras vers la cible : on aligne le bras sur la direction de la cible pour bloquer un angle ( $\gamma$ ).
2. Calcul de la distance cible et du vecteur direction : Maintenant que nous travaillons sur un plan en 2D, on prends la distance vers la cible.
3. Calcul de l'angle  $\theta$ .
4. Calcul de l'angle  $\alpha$ .
5. Calcul de l'angle  $\beta$ .
6. Appliquer les angles aux joints : On applique  $\alpha$  et  $\beta$  aux joints 1 et 2 du bras, et on utilise  $\theta$  pour orienter le bras correctement



# Trajectoire

Deux fonctions principales pour générer nos splines naturelles :

- `CalculateCoefficients()` : Résolution d'un système d'équations pour chacun des segments de notre spline, 4 coefficients pour X et 4 pour Y. On récupère les points de contrôle, et on calcul respectivement les coefficients pour X et pour Y.
- `EvaluateSpline()` : Cette fonction va nous permettre de retrouver notre position sur la spline à chaque instant, et sur chaque segment individuellement, grâce à une variable "t".

Le fonctionnement de notre trajectoire est le suivant : on commence par pré-calculer les coefficients des polynômes cubiques pour chaque segment de la spline, pour qu'on puisse ensuite les utiliser pour se retrouver sur la spline avec `EvaluateSpline()`.



**Merci pour votre attention**