Animation d'un robot

IFT3355 : Infographie - TP #1

Jérémie Dumas
Baptiste De La Robertie

3 février 2010

Université de Montréal

Table des matières

Introduction au problème

Le sujet de notre travail consiste en l'animation simple d'un robot, lequel évolue dans un labyrinthe relativement basique. Le problème s'axait notamment sur le calcul des transformations nécessaires pour passer du repère local d'un objet (un bras du robot par exemple) au repère de son parent (torse), jusqu'au repère global de la scène.

Le reste du devoir contient également deux autres parties : animation du robot, avec une marche simple mais réaliste (on évite le footskating et on adapte la hauteur du robot), et enfin détection des collisions avec le labyrinthe, et application d'une correction appropriée. Les paramètres de la scène, tel que les éléments qui composent le robot, ou les touches pour se déplacer, sont rappelées dans le sujet originel, aussi nous ne reviendrons pas là dessus.

1 Transformations

1.1 Passage d'un repère à un autre

Il y a plusieurs façons de composer des matrices pour obtenir une transformation du repère local d'un objet au repère local de son parent, notamment en ce qui concerne les rotations. Par exemple, pour définir une matrice de rotation R, on peut composer de manière *intrinsèque* des matrices de rotations $R_z(\Psi)$, $R_x(\theta)$, $R_y(\phi)$ autour des axes du repère, en gardant à l'esprit que les axes "bougent" au fur et à mesure qu'on applique les rotations. Les angles ψ, θ et ϕ correspondent alors aux angles d'Euler.

On peut également composer les rotations de manière *extrinsèque*, c'est à dire en fonction des directions x', y' et z' du repère original (cf. figure 1). Il a aussi été choisit d'exprimer la translation $\vec{t}$ dans le repère R_1 (i.e *rotaté* mais non *scalé*). On a donc au final une matrice de transformation $H = ST(\vec{t})R(\alpha, \beta, \gamma)$, où α, β, γ désignent les rotations autour des axes x', y' et z' du repère $\mathcal{R}'$. Pour passer d'un point $P_{\mathcal{R}'}$ exprimé dans $\mathcal{R}'$ à un point $P_{\mathcal{R}}$ exprimé dans $\mathcal{R}$, on fait donc $P_{\mathcal{R}} = H \cdot P_{\mathcal{R}'}$ (en coordonnées homogènes).

Le choix a été fait d'appliquer d'abord la rotation, car cela facilite l'animation du robot, que ce soit pour la marche ou pour la rotation du bras (le centre de rotation coïncidant alors avec le centre du repère).

1.2 Assemblage des éléments

Le robot Il est constitué de cylindres, boîtes et sphère, sa géométrie est donc relativement simple. Il convient cependant de créer correctement sa hiérarchie, et de positionner chaque élément par rapport à son parent. Le choix d'appliquer des rotations extrinsèques en premier nous a bien aidé à positionner chaque objet sans trop de soucis. Notons notamment que le torse est orienté vers le haut, et que les bras et les jambes sont au début orientés vers le bas (axe des y vers le bas).

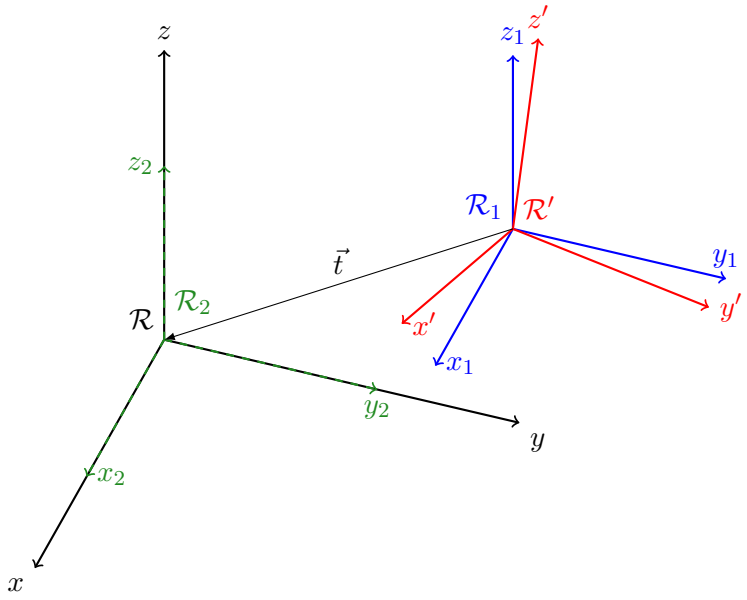


FIGURE 1: Exemple de passage du repère local d'un objet ($\mathcal{R}'$) à celui de son parent ($\mathcal{R}$).

Le labyrinthe Pas de grandes difficultés pour placer les murs du labyrinthe ici. Si A et B sont deux points du segment définissant un mur, on crée d'abord une boîte canonique de longueur AB définie dans son repère $\mathcal{R}'$ local. On calcule ensuite la translation qui amène l'origine de $\mathcal{R}'$ sur le point A. Reste alors à trouver la rotation à appliquer, ce qui se fait en récupérant le cosinus du vecteur unitaire associé à $\vec{AB}$ (en faisant attention au signe).

2 Animation du robot

2.1 Contrôles simples

Rotation du torse C'est l'animation la plus simple qui soit : il s'agit d'appliquer une rotation sur le nœud `m_torso` autour de l'axe des Z de son repère $\mathcal{R}'$ local (axe qui pointe vers le haut dans notre cas). Le code actuel effectue un modulo pour ramener l'angle que fait le repère $\mathcal{R}'$ avec son parent entre 0 et 2π (pour éviter les overflow), mais on aurait pu s'en passer.

Mouvement des bras et avant-bras Pas de difficulté là non plus : on applique autour de l'axe X' du repère local une rotation d'un angle α proportionnelle à la différentielle de temps Δt depuis le dernier rendu. On ne borne cependant pas les angles que peuvent faire les éléments, mais on peut vouloir limiter le mouvement de l'avant-bras à une rotation entre 0 et π par exemple.

2.2 Contrôle de la marche

2.2.1 Cadre général

Il y a plusieurs choix à faire pour l'animation. Pour simplifier notre explication, on se place dans le cas d'une marche d'un robot dans le plan Oyz, l'axe Oy étant l'horizontal.

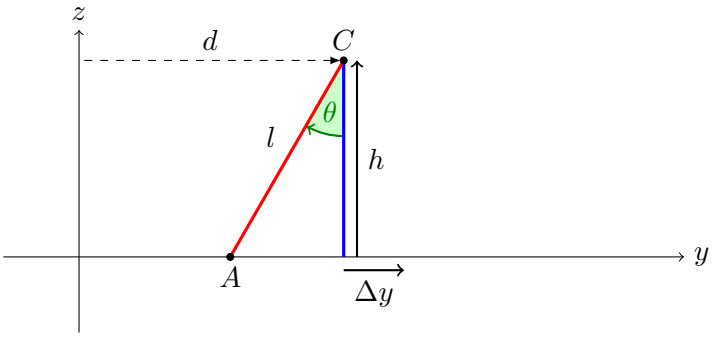
On a plusieurs paramètres reliés entre eux pour assurer la cohérence de la marche du robot (cf. schéma de la figure 2) :

- θ : L'angle que fait la jambe gauche avec la verticale Oz.
- d : La position sur l'axe Oy du centre C du robot (distance parcourue).
- h : La position sur l'axe Oz du centre C du robot (hauteur).

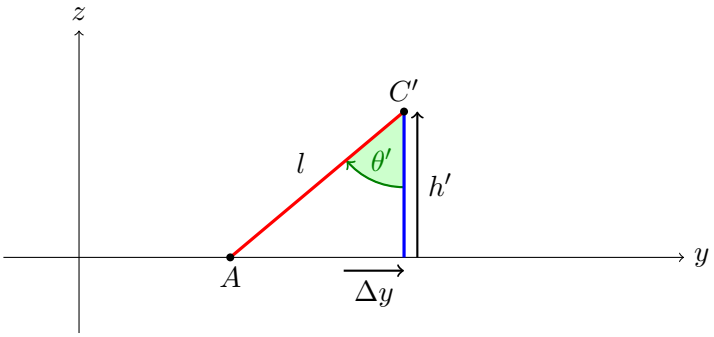
Variantes possibles

- Faire avancer le robot à vitesse constante : $d(t) = v \cdot t$ (solution adoptée)
- Animer l'angle θ d'un mouvement sinusoïdal : $\theta(t) = \theta_{max} \sin(2\pi \frac{t}{T})$
- Faire varier θ de manière uniforme, ce qui nous donnerait une fonction $\theta(t)$ périodique de période $4T$ de la forme :

$$\theta(t) = \begin{cases} \theta_{max} \frac{t}{T} & \text{si } t \in [-T, T] \\ -\theta_{max} \frac{t-T}{T} + \theta_{max} & \text{si } t \in [T, 3T] \end{cases}$$



(a) Avant le déplacement



(b) Après le déplacement

FIGURE 2: Animation de la marche du robot.

2.2.2 Correction de la hauteur

On a un déplacement $\Delta y = v \cdot \Delta t$. En observant la figure 2 et en faisant un peu de maths, on peut établir les relations suivantes qui donnent θ' et h' en fonction du déplacement :

$$\sin(\theta') = \sin(\theta) + \Delta y$$
$$h' = \sqrt{l^2 - l^2 \sin(\theta')^2}$$

On fait attention à seuiller pour garder θ entre $-\theta_{max}$ et θ_{max}. On ajoute ensuite une correction supplémentaire à h qui prend en considération le fait que le pied soit une boîte : on fait en sorte que seul le talon du pied d'appui touche bien le sol. Cependant si on y regarde de près, cela signifie que le point d'appui pour la rotation de la jambe (i.e. le centre du pied, qui n'est pas en contact avec le sol), a une coordonnée en y qui ne bouge pas, alors que le point de contact avec le sol (i.e. le talon du pied), va lui bouger légèrement selon y.

3 Détection des collisions

3.1 Collisions avec les murs

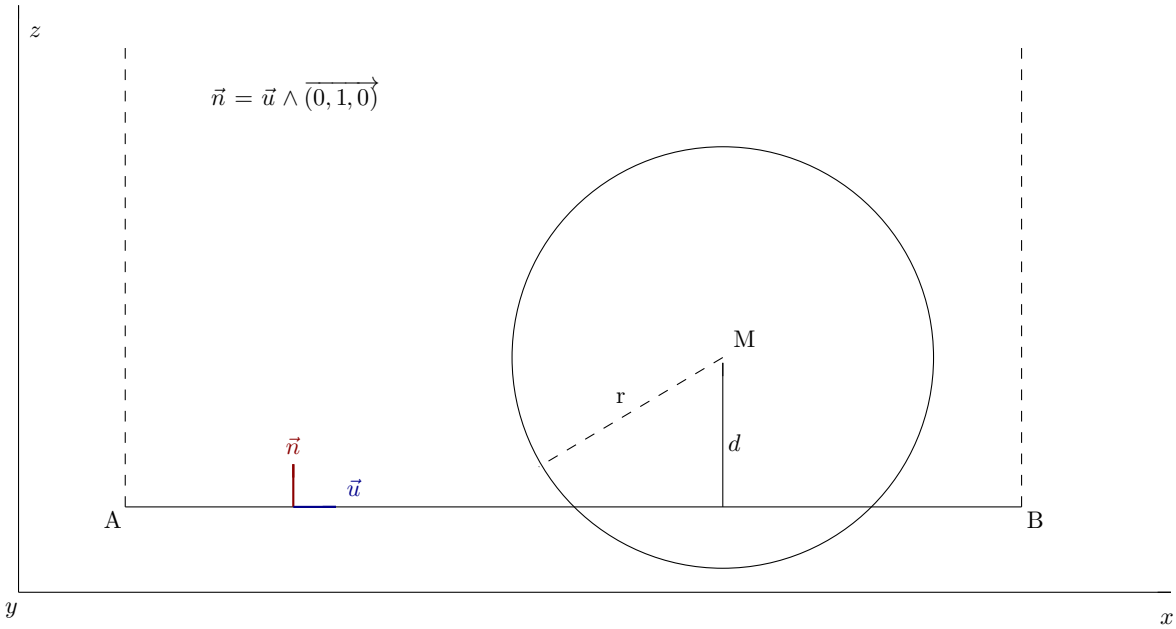
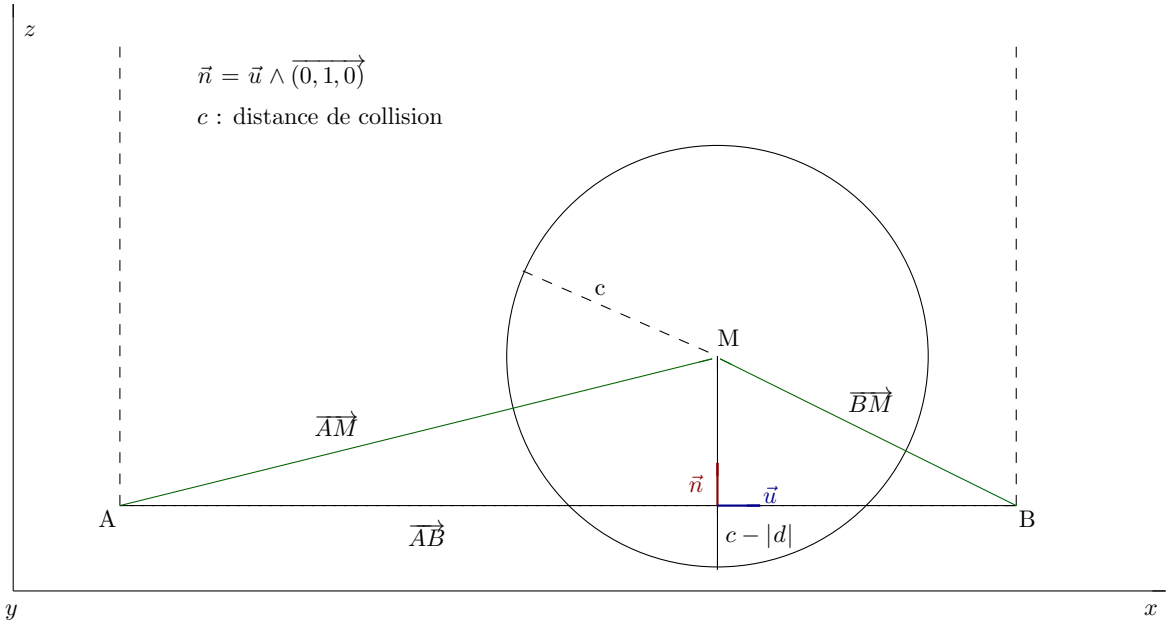


FIGURE 3: Collision générique entre le robot et un mur du labyrinthe.

On se place dans le repère local du labyrinthe. On cherche à déterminer si le robot, situé en M, entre en collision avec un mur du labyrinthe, défini par deux points A et B (cf. figure 3). On peut différencier deux cas de figure :

Cas intérieur La projection de M sur la droite (AB) se situe sur le segment $[AB]$. Cela se produit lorsque $(\overrightarrow{AM} \cdot \overrightarrow{AB}) \cdot (\overrightarrow{BM} \cdot \overrightarrow{BA}) < 0$ (cf. figure 4).



d

FIGURE 4: Cas où la projection du robot est sur le segment $[AB]$.

Nous calculons dans ce cas la distance d signée entre le point M et la droite (AB). Si $\vec{u}$ est le vecteur unitaire associé à $\overrightarrow{AB}$, et que (AB) se situe dans le plan Oxz, on peut obtenir cette distance d en calculant :

$$d = (\vec{u} \wedge \overrightarrow{(0,1,0)}) \cdot \overrightarrow{AM}$$

Selon le signe de d, nous appliquons alors au robot une translation d'intensité $c - |d|$ dans le sens opposé au mur, où c est la distance de collision (rayon du cylindre représentant le robot).

Cas extérieur Se produit lorsque le robot est situé à l'extérieur du bandeau formé par les extrémités du segment (cf. figure 5). Il suffit alors de considérer le point le plus proche du robot (entre A et B), et d'effectuer une translation dans la direction opposée à celle du robot.

Mettons que $d = \|\overrightarrow{AM}\|$ désigne une telle plus petite distance, on effectue alors une translation de facteur $c - d$ dans la direction $\overrightarrow{MA}$ (resp. $\overrightarrow{MB}$).

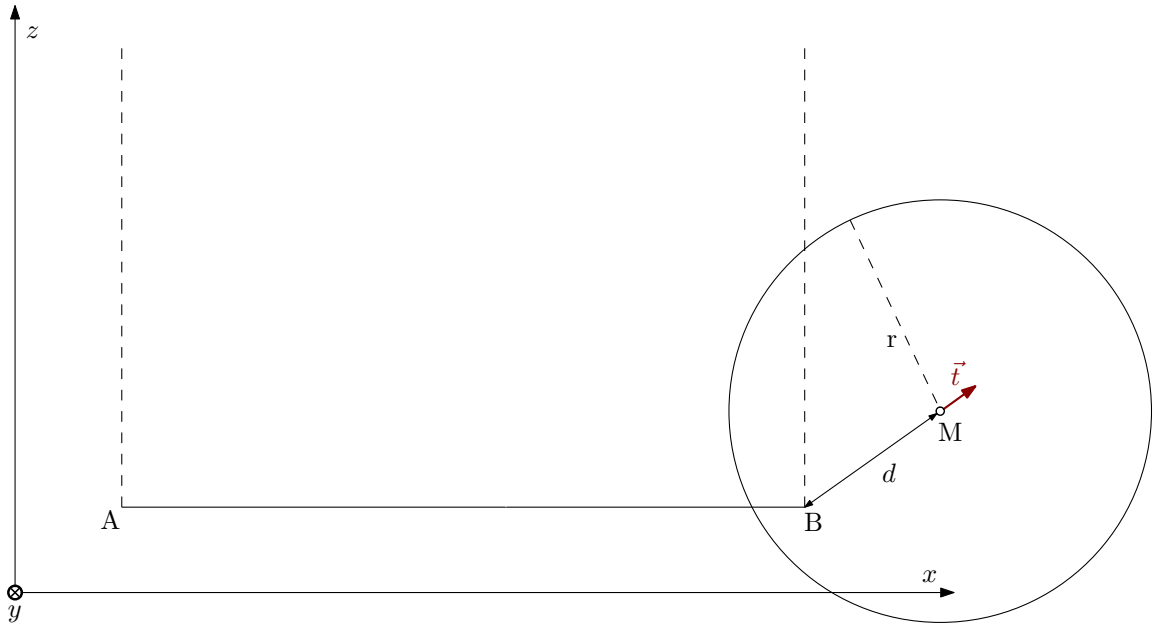


FIGURE 5: Cas où la projection du robot est en dehors du segment $[AB]$

Avantages, inconvénients Il convient bien entendu, dans les deux cas précédant, d'exprimer la translation à appliquer dans le repère local du robot avant d'appliquer la transformation.

Un autre point important auquel il faut faire attention, c'est l'invariance aux changements d'échelle et rotation. Actuellement, le code résiste bien aux rotations qui peuvent être appliquées au labyrinthe par exemple. Cependant, les distances enregistrées sont exprimées dans le repère local cible (repère du labyrinthe par exemple). Si on fait un scaling du robot par s, il faudra donc penser à multiplier notre attribut par s.

En pratique l'idéal serait, une fois qu'on a calculé la transformation H pour passer du repère $\mathcal{R}_1$ du robot au repère $\mathcal{R}_2$ du labyrinthe, d'en déduire le facteur d'échelle correspondant. On pourrait faire ça en mesurant le ratio $\frac{\|\overrightarrow{A'B'}\|}{\|\overrightarrow{AB}\|}$ où A, B sont deux points de $\mathcal{R}_1$, et $A' = HA$, $B' = HB$ sont leurs images dans $\mathcal{R}_2$.

3.2 Toucher la cible

Cette partie a sans doute été la plus aisée à implémenter. Il a simplement été question de déterminer quand un point de la main du robot se trouvait à l'intérieur de la cible, exprimées dans le repère global. Plus formellement, nous nous contentons de calculer la distance entre un point M de la main et le centre de la sphère, dont les coordonnées globales sont connues. Pour plus de précisions, le point M a été choisi sur la face "avant" de la main du robot.

Conclusion

Finalement on remarque que l'implantation effectuée nous a permis de mettre en lumière plusieurs choses. Tout d'abord, il n'y a pas de façon unique de procéder. On peut effectuer les transformations entre repères de multiples façons, il convient de bien spécifier quel argument correspond à quoi, et d'en rester conscient lorsqu'on place nos objets dans la scène.

De même pour la marche, nous avons suggéré plusieurs méthodes pour animer le robot de manière simple. Il y en a des plus ou moins réalistes, mais il est difficile de départager. Enfin il existe bien des modèles plus complexes pour faire marcher un personnage (en pliant les genoux par exemple), qui sont déjà bien plus réalistes, mais plus compliqués à mettre en équation.