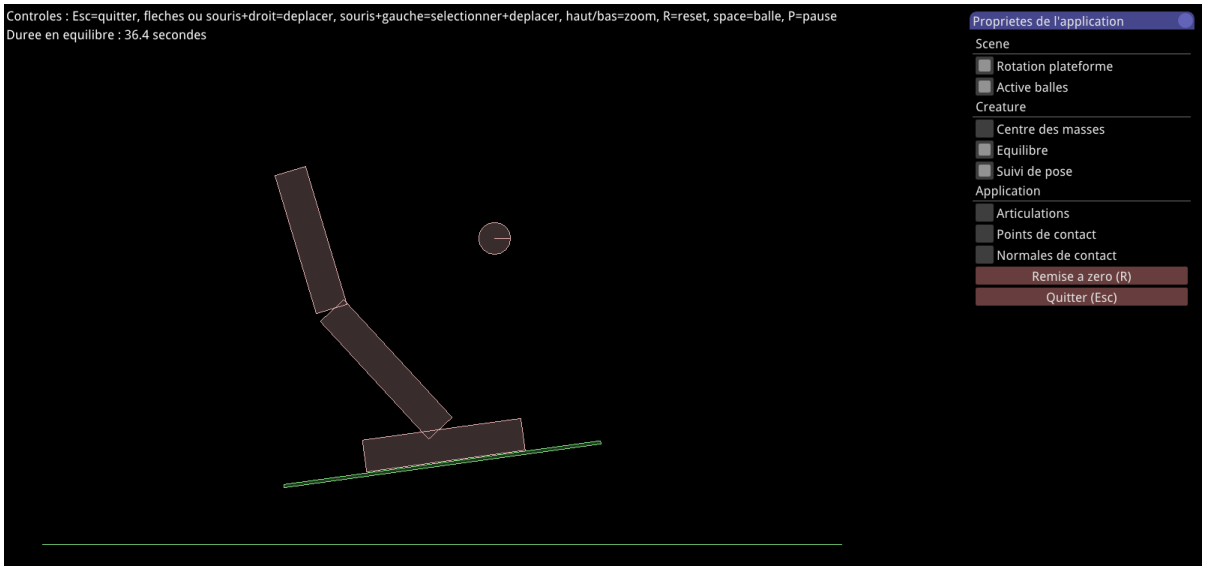
M1if37 Animation en synthèse d'image

TP contrôleur de mouvement

Objectif du TP : Contrôler le mouvement d'une créature simple en 2D avec un moteur physique



Description courte : Ce TP est réalisé individuellement. Il repose sur les principes introduits en cours (animation de squelette, principe physique, et contrôleur). Il est composé de 2 parties, une partie de « prise en main » du simulateur et une partie « développement ». Votre TP sera évalué sur la base du code fourni, d'un rapport à rédiger et d'une présentation orale de quelques minutes.

Préparation : Téléchargez l'archive correspondant au TP contrôleur sur le site de l'UE. Cette archive contient :

- un répertoire **`Box2D`** contenant les librairies, les fichiers sources, des exemples, un manuel et les projets CodeBlocks permettant de compiler les librairies du moteur physique Box2D
- un répertoire **`src`** contenant les classes du simulateur
- un projet CodeBlocks (Windows et Linux) et un makefile (Linux) pour compiler et exécuter le programme
- des fichiers utiles à l'exécution (ex. un fichier mouvement)

Il est conseillé de travailler sous Windows car tout est déjà précompilé. Si vous souhaitez travailler sur votre machine sous Linux, vous devrez soit installer CodeBlocks et les librairies GLFW et GLEW vous-même (commande : `apt install codeblocks libglfw3-dev libglew-dev`) et compiler les libraires Box2D et IMGUI grâce aux projets CodeBlocks fournis, soit utiliser le makefile fourni.

En compilant et exécutant le projet TP_CONTROLEUR fourni, vous verrez la créature tenir en équilibre sur une plateforme.

Sur la droite de l'application se trouve un menu (que vous pouvez étendre dans le fichier **`Framework/Main.cpp fonction sInterface`**, les valeurs par défaut des options étant définies dans **`Framework/Application.h structure Settings`**). Dans ce menu, vous pouvez activer/désactiver :

- la rotation de la plateforme sur laquelle se trouve la créature
- le lancement de balles sur la créature
- la visualisation du centre de masse global de la créature
- la gestion de l'équilibre
- le suivi de mouvement
- la visualisation d'autres objets (articulations, points et normales de contact)

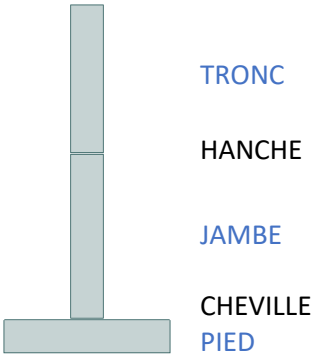
Vous pouvez également réinitialiser la simulation ou quitter l'application.

Partie « prise en main »

1. Maintien de l'équilibre

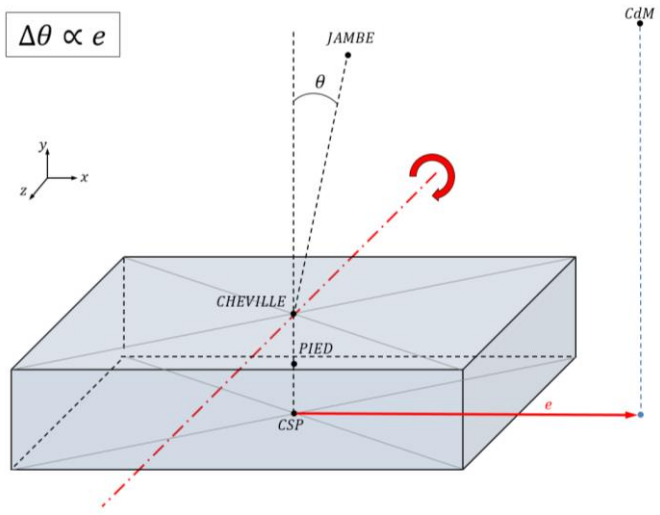
La créature est composée de trois corps rigides (pied, jambe et tronc) connectés par deux articulations à un degré de liberté de rotation chacun (cheville et hanche). Le monde physique est en 2D, les axes de rotation sont donc orthogonaux à cet espace 2D.

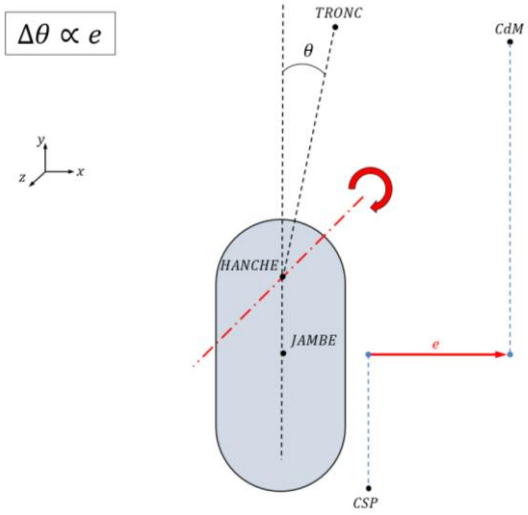
- Le tronc est une boite de 40 cm de largeur et 1.8 m de hauteur pesant 3.5 kg
- La jambe est une boite de 40 cm de largeur et 2 m de hauteur pesant 4 kg
- Le pied est une boite de 2 m de largeur et 40 cm de hauteur pesant 6 kg



Le premier objectif du contrôleur est de maintenir l'équilibre de la créature indéfiniment (ou jusqu'à ce qu'elle glisse de la plateforme si cette dernière est trop penchée). La stratégie choisie utilise la projection au sol de la position du centre des masses (CdM) de la créature. Le principe général est le suivant (implémenté dans la procédure **`Creature::balanceController`** et illustré par les deux figures ci-dessous) :

1. La position du CdM est calculée en fonction de la position et de la masse de chaque corps rigide (voir fonction **`Creature::computeCenterOfMass`**)
2. La position du centre du polygone de support (CPS) est calculée comme étant la position dans le monde du milieu du dessous du pied
3. L'erreur e entre les deux positions projetées au sol est calculée et est considérée comme proportionnelle à la force à appliquer au CdM pour revenir à l'équilibre
4. La force est convertie en moments articulaires par la méthode de la transposée de Jacobienne (voir fonction **`Creature::jacobianTranspose`**)
5. Les moments articulaires sont appliqués, après éventuelle mise à l'échelle





Etudiez le code de la classe **`Creature`** et en particulier les trois fonctions citées.

2. Suivi de mouvement

En plus de maintenir son équilibre, la créature doit pouvoir suivre un mouvement prédéfini. Par défaut, le mouvement à suivre est donné dans le fichier **`motion.txt`**. Si vous le souhaitez, vous pouvez l'éditer ou créer de nouveaux fichiers de mouvement. La syntaxe est simple. La première ligne du fichier indique, dans cet ordre, le nombre de frames, la fréquence et le nombre de degrés de liberté. Le reste du fichier contient sur chaque ligne les valeurs des angles, en radians, pour chaque frame. La lecture du fichier de mouvement est gérée par la classe **`Motion`**.

Le principe est le suivant (implémenté dans la fonction **`Creature::motionTracker`**) :

1. Pour chaque degré de liberté, on récupère l'angle désiré en fonction du temps écoulé et l'angle courant depuis le moteur physique
2. Le moment articulaire permettant de réduire l'erreur entre l'angle désiré et l'angle courant est calculé à l'aide d'un régulateur PD (voir classe **`PDController`**)
3. On applique le moment articulaire

Etudiez le code de la classe **`PDController`** et de la fonction citée.

Notez que les gains des régulateurs (à la fin du constructeur de **`Creature`**) et les éventuels facteurs de conversion d'erreur de position en force ont été définis empiriquement. Modifiez leurs valeurs pour comprendre leur influence sur le résultat. Faites en sorte que la créature puisse réagir naturellement aux changements de pose et aux perturbations externes.

Partie « développement »

Dans cette partie, vous devez développer vous-même 3 extensions du contrôleur selon les instructions suivantes.

Extension 1

Vous développerez un contrôleur d'équilibre pour une créature plus complexe. Par exemple la créature pourra avoir des corps supplémentaires le long de la chaîne (ex. une cuisse, une tête), des corps supplémentaires dans une hiérarchie (ex. des bras, des jambes), etc. Un développement simpliste tel que la modification des proportions de la créature (ex. pied plus petit) n'est pas considéré comme valide.

Le contrôleur d'équilibre devra prendre en compte ces nouveaux corps.

Extension 2

Vous développerez un contrôleur pour un nouveau mouvement. Par exemple, la créature devra éviter ou repousser les balles lancées, attraper un objet, se déplacer (ex. saut, marche, course), se battre avec une autre créature, etc.

Le suivi de ce mouvement sera réalisé avec des régulateurs PD qui seront réglés pour un réalisme et une réactivité optimaux. Le mouvement sera soit fixe (ex. issu d'un fichier), soit calculé en temps-réel.

Extension 3

Vous développerez l'interactivité du contrôleur. Vous ajouterez pour cela des commandes utilisateurs (comme ça l'est actuellement pour le lancer de balles). Ces commandes peuvent être très variées comme l'activation de forces externes simulant un phénomène physique ou la modification de l'environnement de simulation, et peuvent dépendre des deux extensions précédentes, comme la commande de la direction ou de la vitesse de déplacement, de l'intensité d'un saut, de l'ajout ou de la suppression d'un corps, de la bascule d'un mouvement à un autre (de manière instantanée ou plus fluide), etc.

Le but est de prouver la robustesse de votre contrôleur à des changements ainsi que sa capacité à réagir aux commandes et à les respecter.

Des extensions simplistes telles que la modification de paramètres comme des gains, des coefficients, des masses ne sont pas considérées comme valide. Il en est de même pour l'activation de forces externes simples comme du vent.

L'originalité est un critère important dans ce travail donc évitez de faire un bipède humanoïde qui marche ou qui saute. Bien évidemment des extensions complexes et des rapports bien rédigés seront d'avantage récompensés.

Rendu du TP : Vous devez soumettre une archive contenant le code (exécutable, sources, librairies, projet CodeBlocks et tous les fichiers additionnels nécessaires à l'exécution de votre application), et un document explicatif (Word ou pdf). L'archive sera à déposer sur TOMUSS avant la date limite de dépôt.

Le document explicatif (entre 3 et 6 pages, hors annexes et figures) décrit les 3 extensions que vous avez choisies, leurs implémentations et les résultats. Le modèle à suivre est disponible sur la page de l'UE. Ne donnez pas de code dans ce document, par contre les algorithmes en pseudocode seront tolérés s'ils aident à la compréhension. Il vous faudra donner des éléments de contexte, expliquer les aspects les plus théoriques, les liens potentiels avec le cours magistral, les références à des articles de recherche si vos extensions s'en inspirent. Il faudra également que le rapport explique ce qui a été fait, l'interface utilisateur finale, vos choix de conception (ex. classes) et d'implémentation (ex. structures de données ou algorithmes utilisés), les résultats obtenus et les améliorations possibles. Vous pouvez illustrer vos propos avec des images tirées de l'application, des tables, des figures, des graphiques etc.

Les critères d'évaluation sont les suivants :

- Originalité
- Difficulté
- Qualité de la conception et des résultats
- Robustesse, réactivité et fiabilité du contrôleur
- Clarté et organisation de la présentation
- Qualité de la documentation du code et du rapport