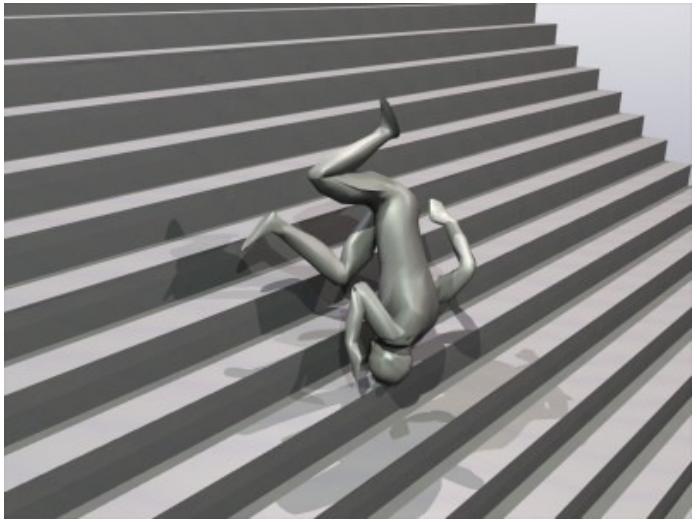
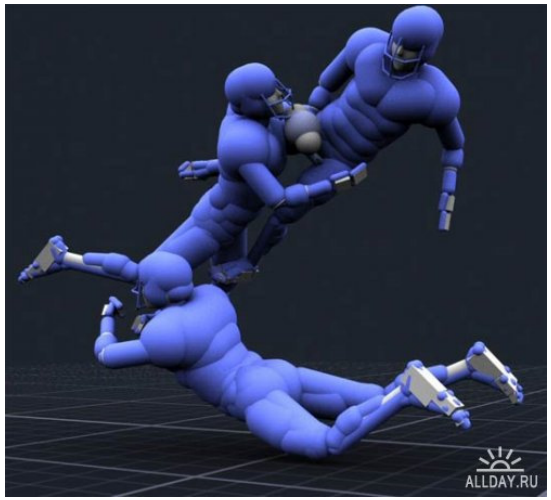


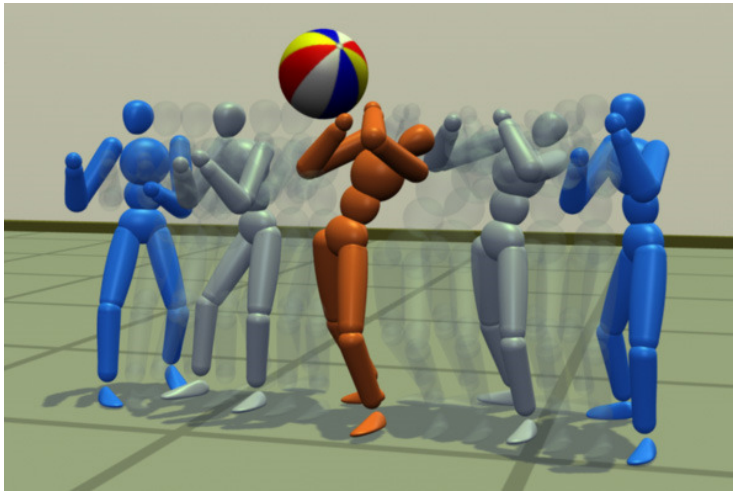


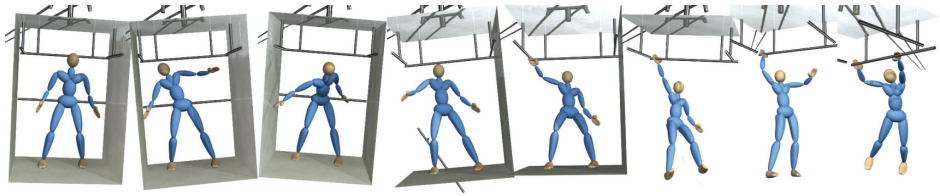
Animation de personnage : motion capture et animation physique

Alexandre Meyer[1]

[1]Equipe SAARA, laboratoire LIRIS

Master ID3D





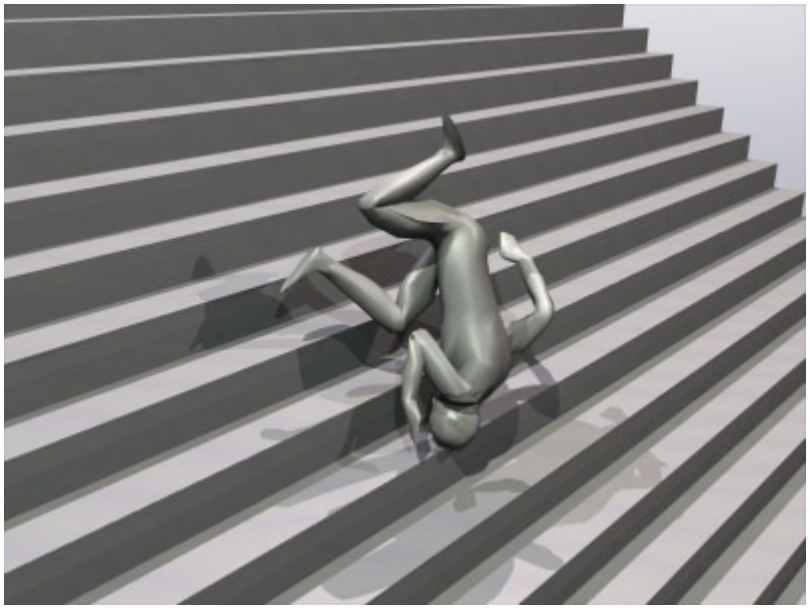
CharAnimation : mocap VS physique

- Mocap
  - + Réalisme
  - - Réalisme pour 1 animation
  - - Système lourd
  - - Edition fastidieuse (souvent intervention humaine)
- Animation physique
  - +Tout automatique
  - - Tout automatique

  ➔ Mélanger les deux!!

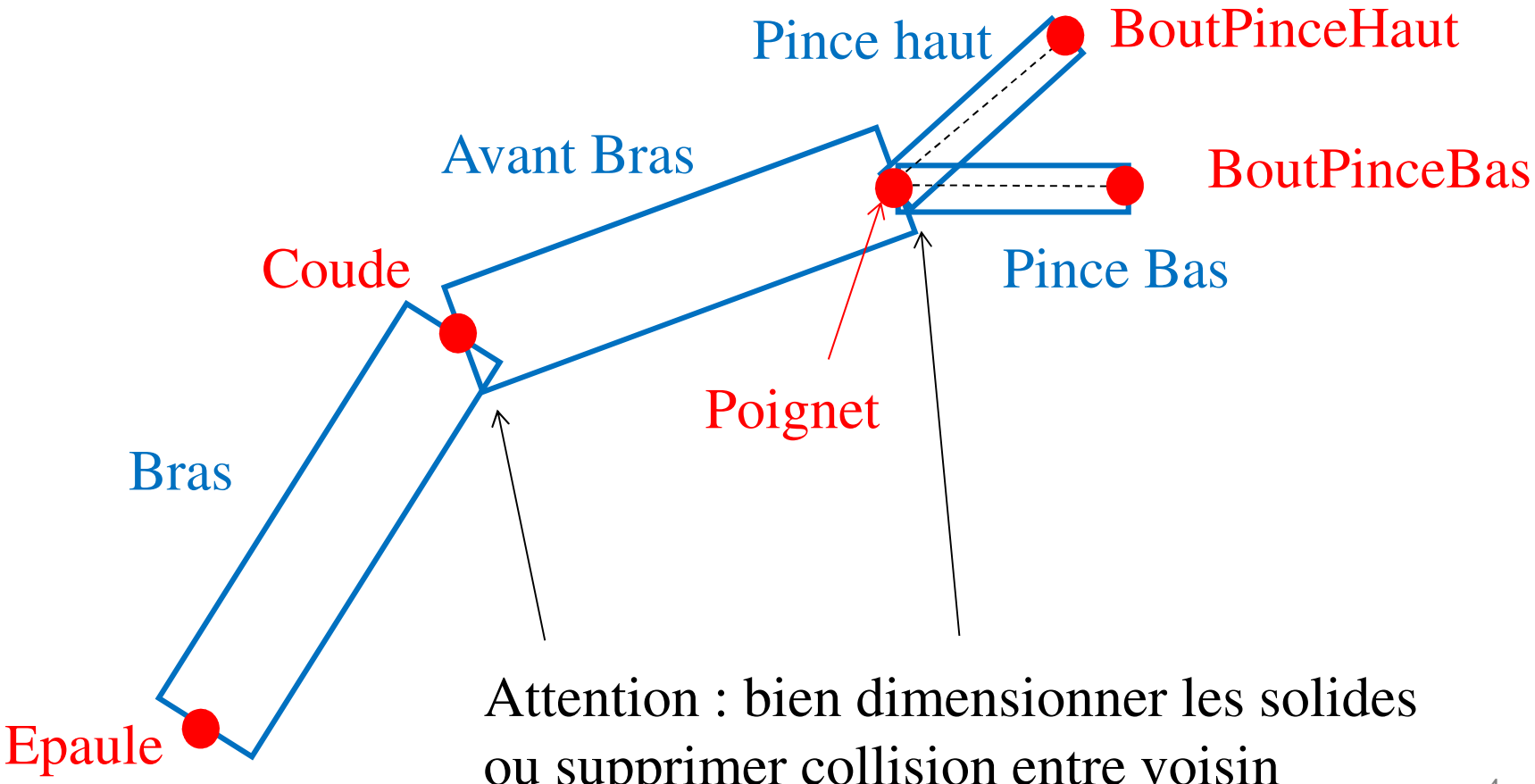
Ragdoll = poupée désarticulée

- Poupée désarticulée
  - Corps = ensemble de membres + articulation
  - Membre = solide rigide (cube, capsule, etc.)
  - Articulation = contrainte de liaison entre les membres
- Vidéo
- Ou démo Bullet
  - http://bulletphysics.org/



Ragdoll = poupée désarticulée

- Articulation = contraintes
- Membre = solide rigide animé



Physique : Newton + contraintes

- Un cycle de calcul physique =
  - Equation physique sur chaque partie du corps
    - Newton : $\sum F = ma$ et $\sum L = I.\omega$
  - Résolution des contraintes
    - Connexion des articulations

➔pour ragdoll globalement 2 méthodes :

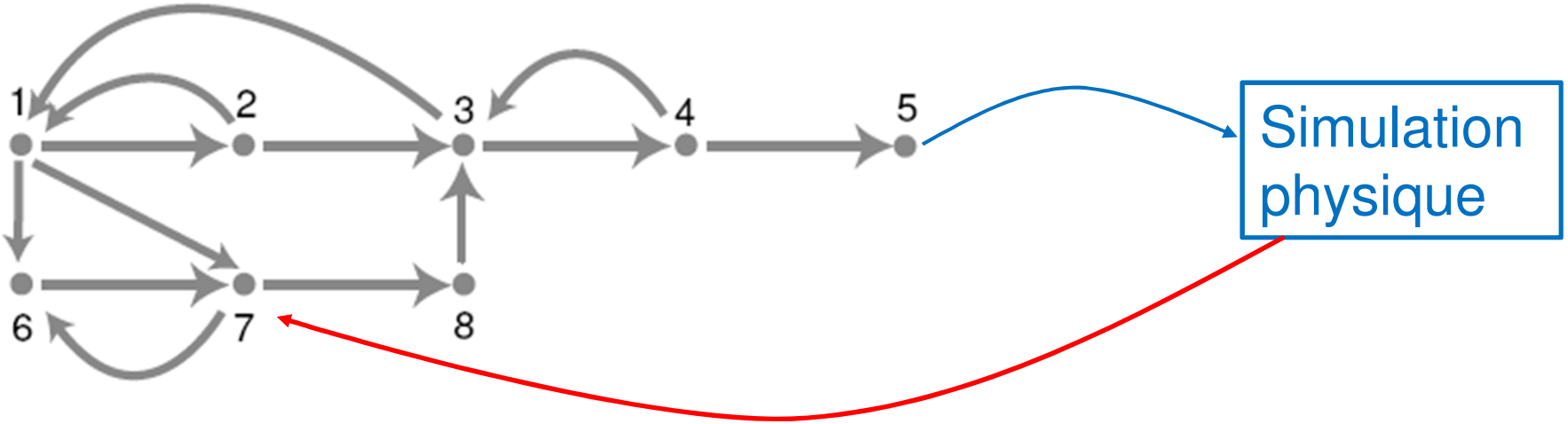
Featherstone, R. (1987). *Robot Dynamics Algorithms*. Kluwer. ISBN 0-89838-230-0.

ou

D. Baraff. Linear-time dynamics using Lagrange multipliers. *SIGGRAPH 1*996

MoCap/Physique : graphe d'animation

- Graphe d'animation + physique
  - Comporte des nœuds de sortie vers la physique
- Utilisable seulement dans certains cas
  - Chute, coups, …



Problème : trouver cette flèche pour réentrer dans le graphe

MoCap/Physique : graphe d'animation

- Graphe d'animation + physique
  - Problème : rentrer dans le graphe après la physique

    ➔ Anticiper quelques frames de physique et chercher les similitudes de positions

    ➔ Ajouter des forces « virtuelles » sur les articulations pour les diriger vers une position du graphe

    ➔ Demande un graphe de MoCap bien remplit avec des séquences pour se relever, rouler, etc.

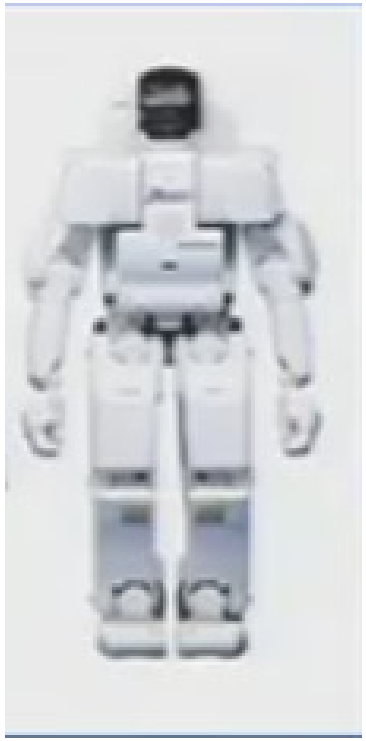
VIDEO

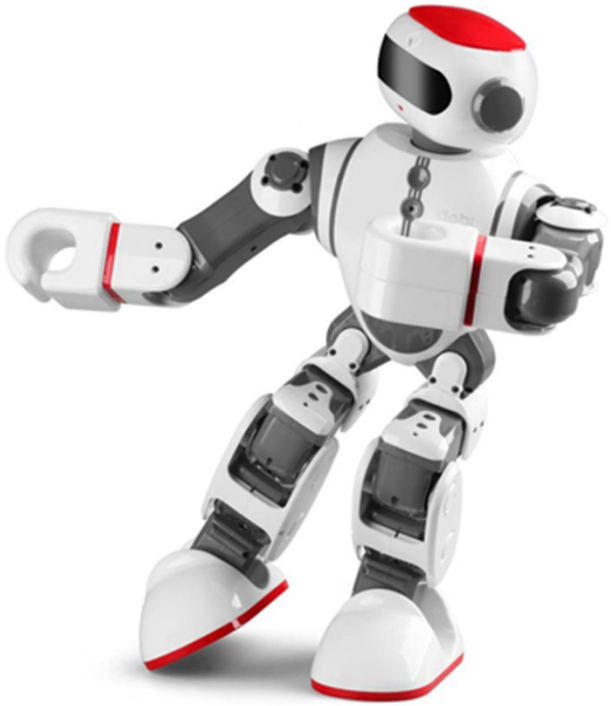
Encore plus loin : tout physique

- Objectifs
  - Animations plus réalistes/réactives : poids, fatigue, etc.
  - Editer les effets physiques : changer gravité, etc.
- Principe
  - On part d'un ragdoll et on essaie de lui donner un contrôle moteur = un « cerveau » dédié à l'animation
  - Un graphe d'animation peu jouer le rôle de « mémoire » de mouvements
- Problèmes
  - Contrôle de l'équilibre (*balance control strategy*)
  - Combiner le mouvement entre la mocap et le réactif
  - …

Controller un personnage physique

- Problème similaire en robotique
  - Incompatibilité entre mocap humaine et robot
  - Avec des problèmes supplémentaires
    - Contrôle/Réactivité des moteurs





Principe : contrôle

- Character Animation using Simulated Physics
- Lié au contrôle

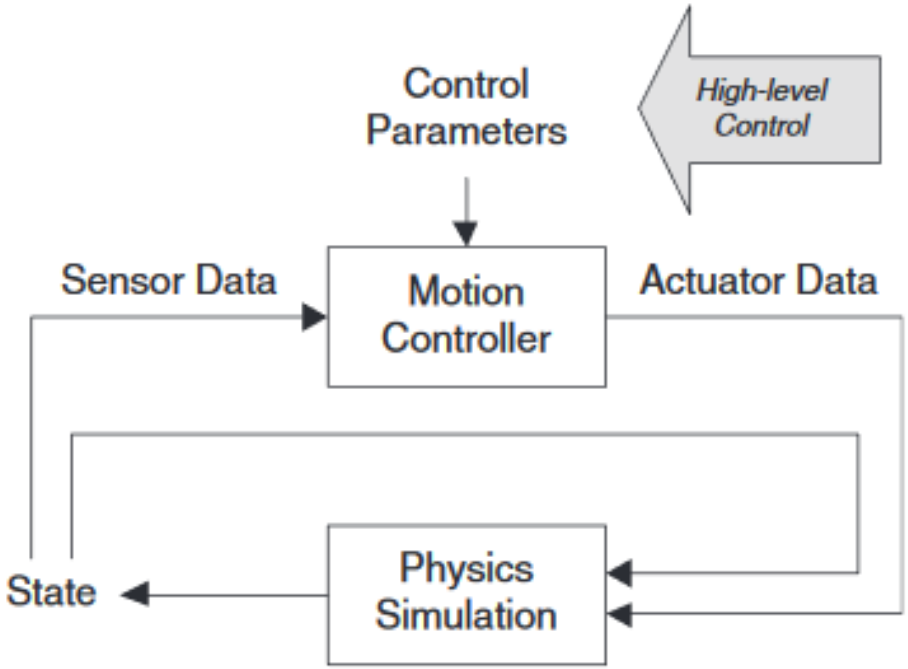


Figure 4: *Closed-loop motion control.*

Controller un personnage physique

Controller un personnage physique

- Motion Controller donne une position cible qu'il estime répondre aux contraintes
  - Equilibre
  - Suivre la trajectoire demandée
- Forces « virtuelles » s'appliquent au ragdoll pour atteindre la position cible
  - Amener chaque articulation vers l'angle désiré
  - Proportional-derivative (PD) control

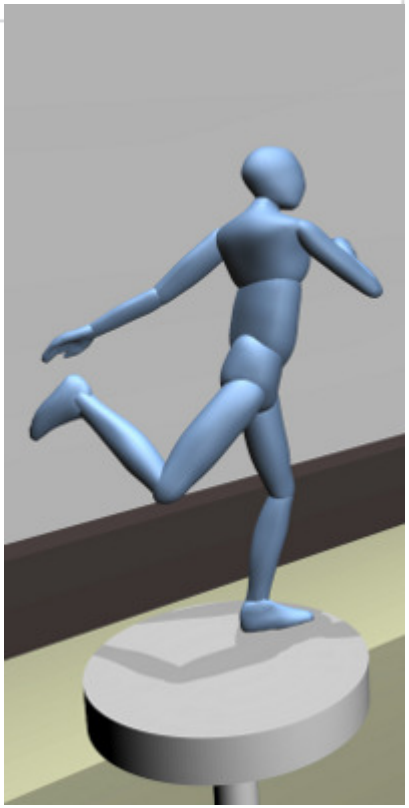
$$\tau = \underbrace{k_p(\theta_d - \theta)}_{\text{Respond to changes.}} - \underbrace{k_d\dot{\theta}}_{\text{Damp}}$$

Tenir debout

- Contrôle de l'équilibre
  → Problème complexe

- Optimisation : Fonction = stabilité
  - Projection du centre de gravité sur le sol
  - Comparer à la position des pieds (support)

  → Fournir une fonction indicateur de stabilité



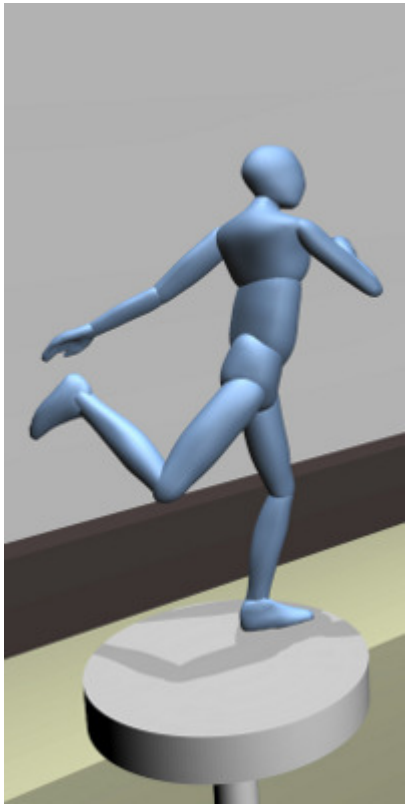
- Dépend des angles entre chaque articulation

  ➔ Non linéaire

Equilibre + machine à état

- Stratégie qui semble marcher [MZS09,JYL09]
  - Optimisation + machine à état

- Machine à état
  - 2 pieds au sol → une fonction
    - Optimisation de la position des bras et du buste pour maintenir l'équilibre
  - Lève un pied → une autre fonction
    - Optimisation des bras, du buste et de la jambe
  - Etc.
- **VIDEO**

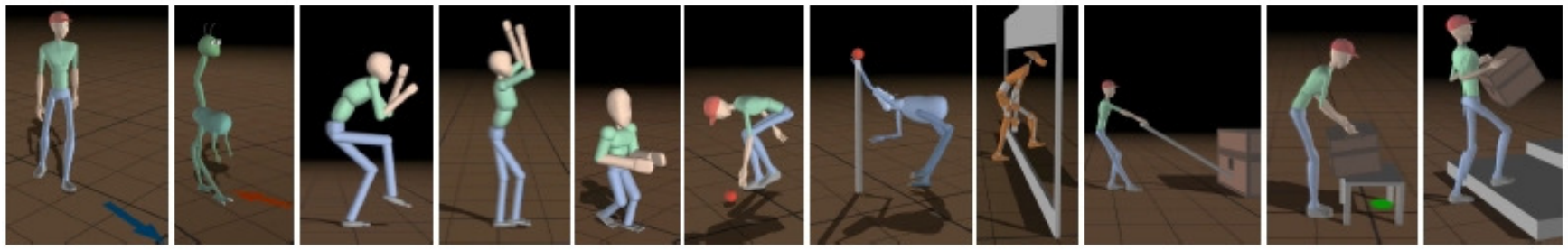


Contrôle : la marche

- Un exemple : SIMBICON (SIMple BIped CONtroler) 2007 (A bien relancé l'idée en recherche)
  - au temps t, prévoir la position cible au temps t+1 avec
    - Fonction erreur inclus l'équilibre et la trajectoire à suivre
    - Machine à état
    - Le Jacobien revient à rendre linéaire la fonction : ok pour un pas de temps

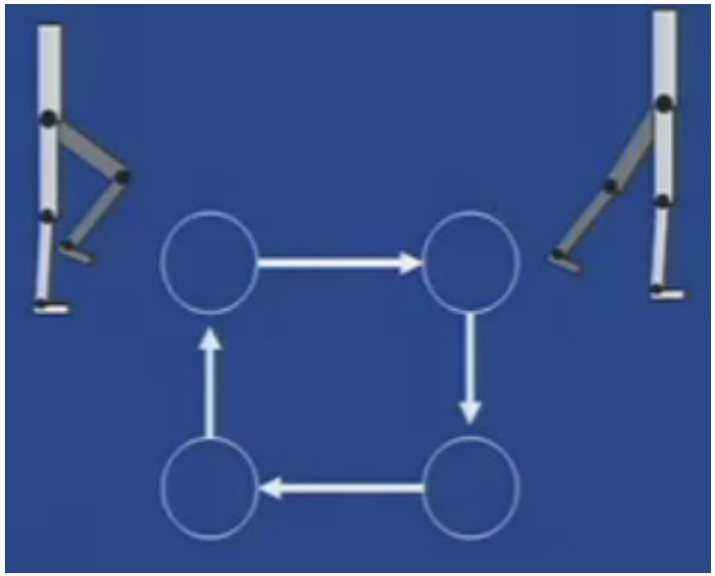
  + Changement de morphologie/objet en temps réel

  - Animation très robotique

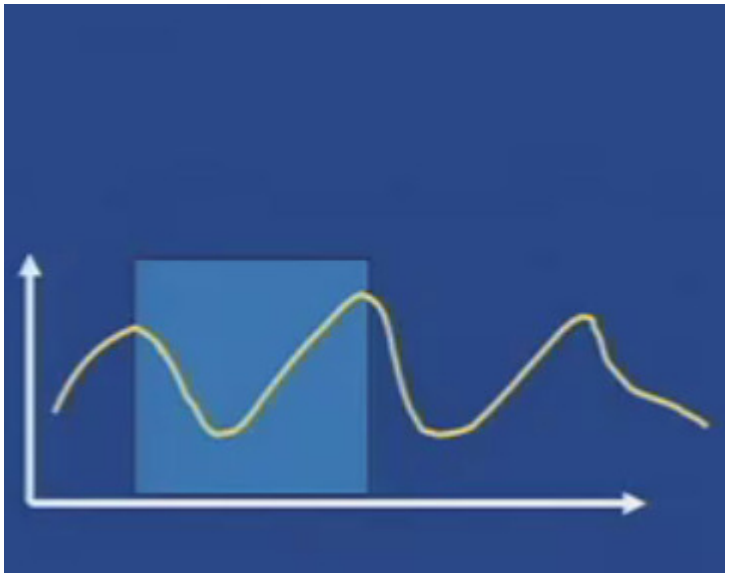


Marche : Machine à états finis

- Marche cyclique



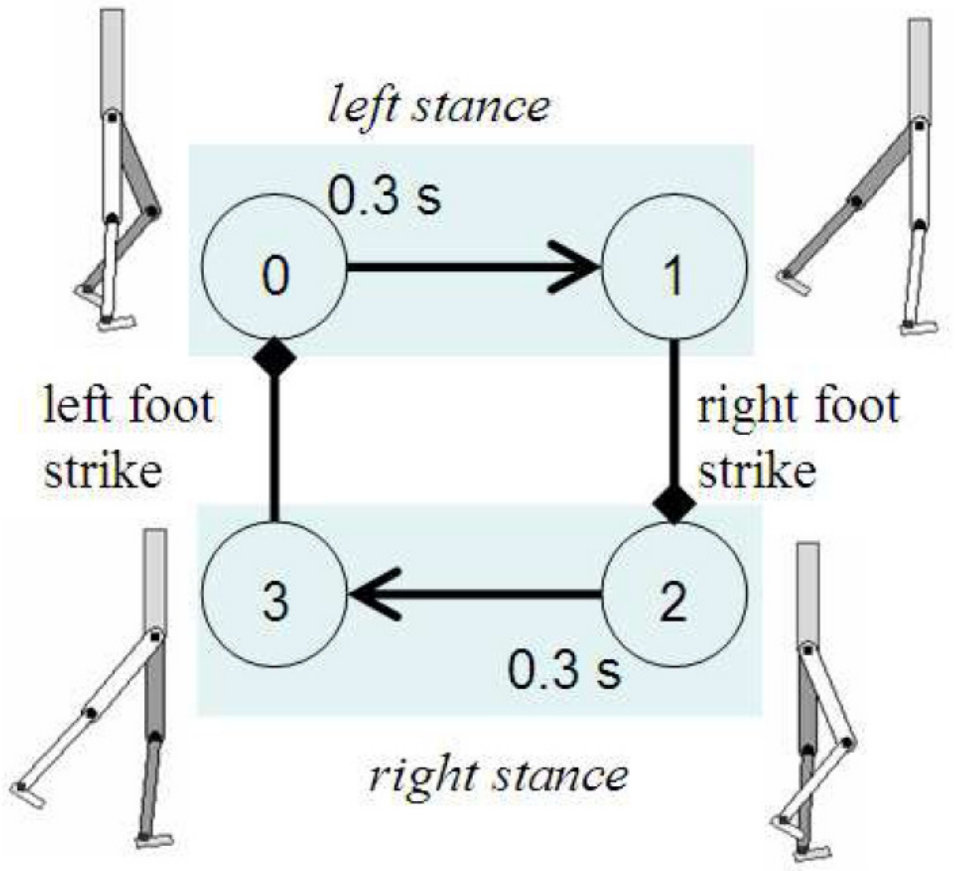
Automate Machine à états finis
(FSM : Finite State Machines)



Motion capture

Marche : machine à états finis

- 4 états :



state 0 → state1
: after 0.3s

state 1 → state 2
: after swing foot contact

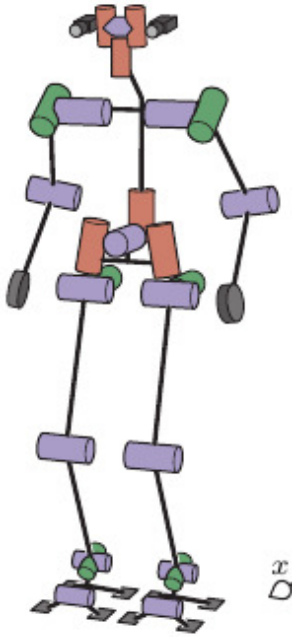
(SIMBICON) … VIDEO

Optimisation

Optimisation de la fonction d'erreur

- Un humain avec 10 à 15 articulations
- Minimum ~15/30 paramètres par pose
- Un clip de 2 ou 3s à minimum 10 poses /seconde

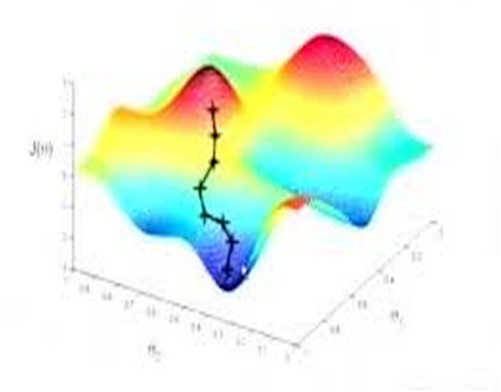
➔ 300 paramètres à trouver

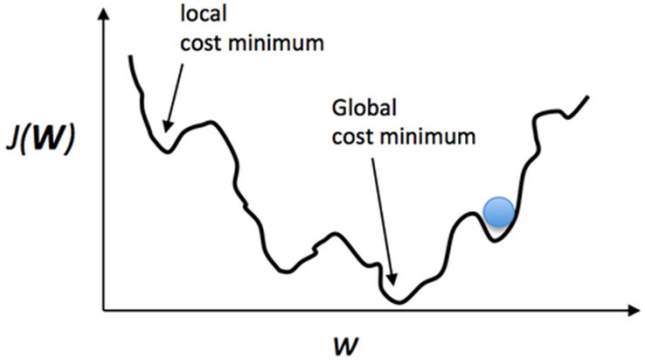


Joint	DoF
Head	3
Shoulder	2
Elbow	1
Pelvis	2
Hip	3
Knee	1
Ankle	2
Toe	1
Total	**25**

Optimisation/temps réel

- Optimisation de la fonction d'erreur
  - Jacobien
    - Linéarise : ok pour t+1 et si fonction sans minimums locaux
    - Temps réel

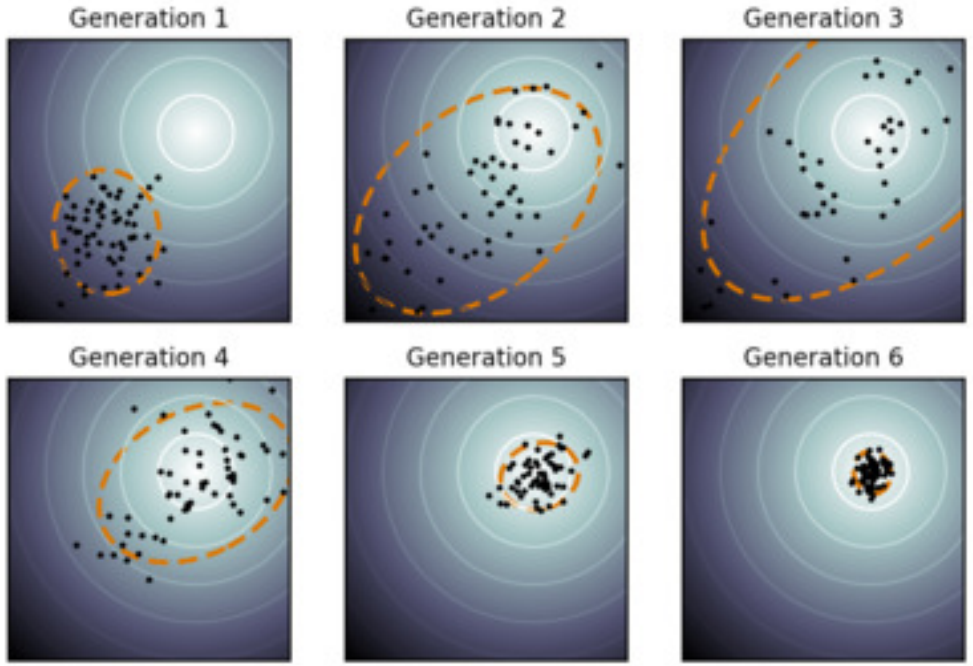




- Dépend de votre point de départ
  - Si MoCap en entrée, l'optimisation sera plus rapide
    - Le mouvement ressemblera à la MoCap
  - Si Random en entrée, il faut explorer plus longtemps

Optimisation

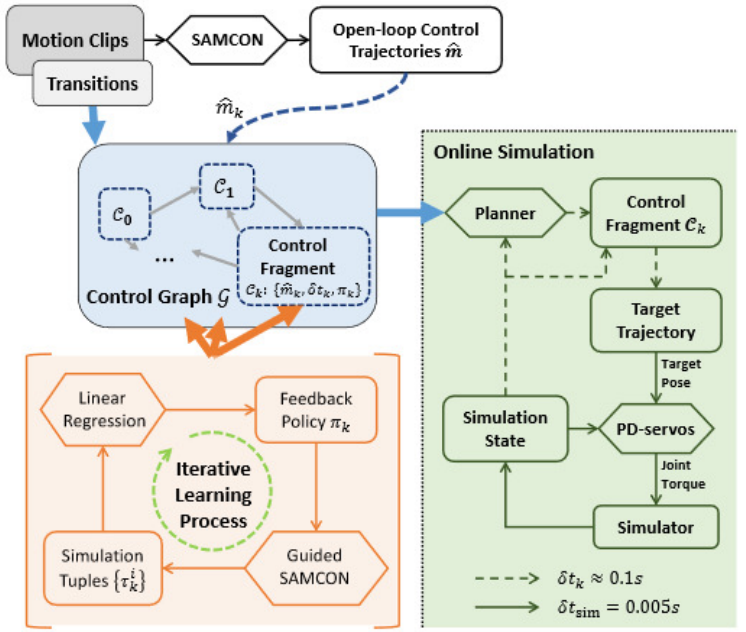
- Optimisation de la fonction d'erreur
  - CMA-ES
    - Covariance Matrix Adaptation Evolution Strategy
    - Passe par-dessus des minimums locaux
    - Difficile en temps réel

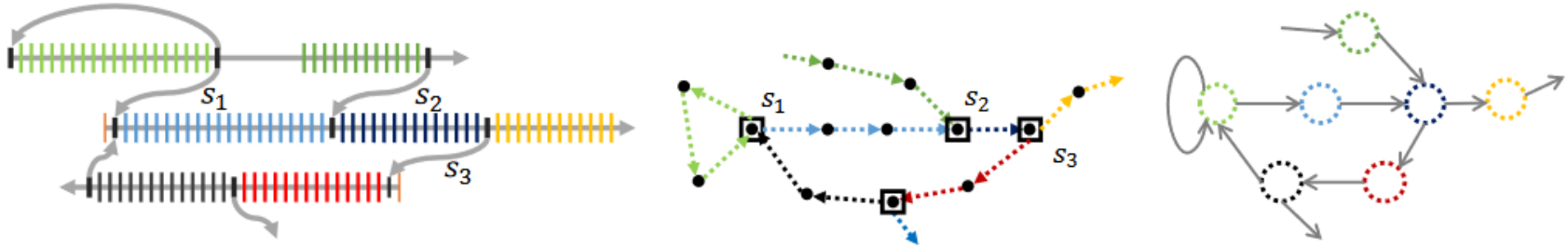


Graphe d'animation et physique

Guided Learning of Control Graphs for Physics-based Characters (2016)

+ VIDEO



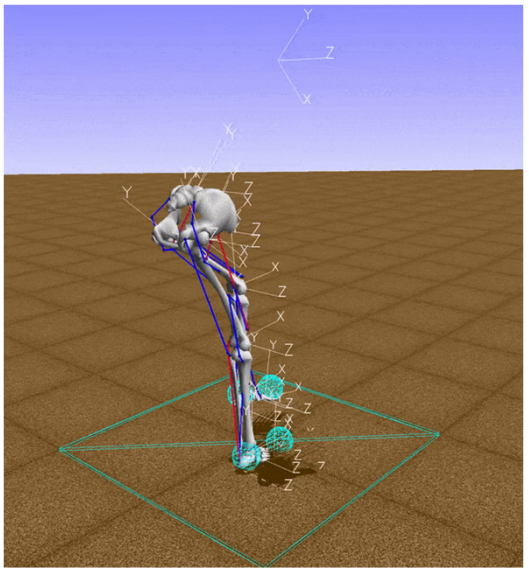


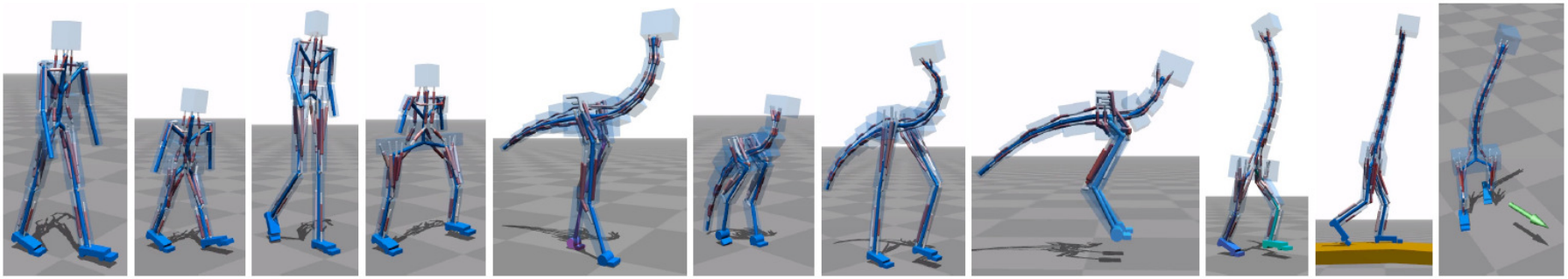
(a) a motion graph (b) a control graph (c) a compact representation of (b)

Fig. 4: Control graph: a control graph is created by (a) building a reference motion graph from example motion clips, then (b) converting each clip of the motion graph to a chain of control fragments. (c) shows a compact representation of the control graph (b), where each node represent a chain of control fragments, or rather, a controller.

La qualité de la simulation

- Simulation physique a de l'importance
  - Pour faire mieux que marche robotique
  - Morphologie humaine, tendons, muscles
    - NIPS 2017 : learning to run, plateforme avec OpenSIM : https://github.com/stanfordnmbl/osim-rl

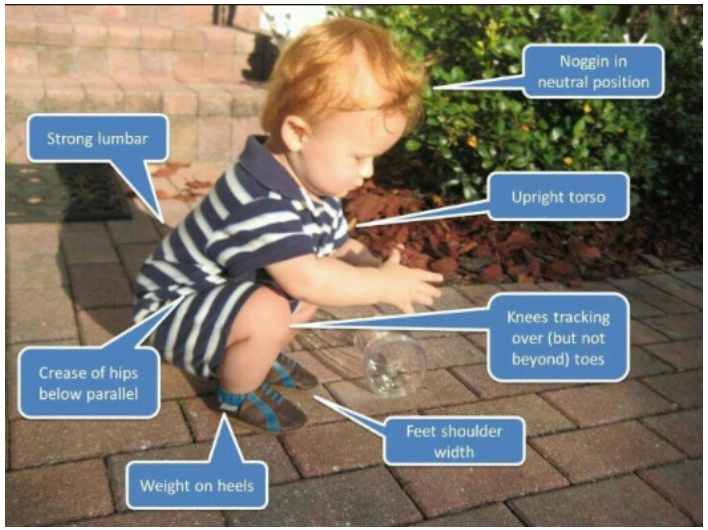




Optimisation VS Apprentissage

- Optimisation
  - Trouve une unique animation pour une fonction d'erreur donnée et précise
  - Exemple de fonction d'erreur
    - Tenir debout
    - Suivre la trajectoire voulues
  - Peut prendre plusieurs h de calculs pour

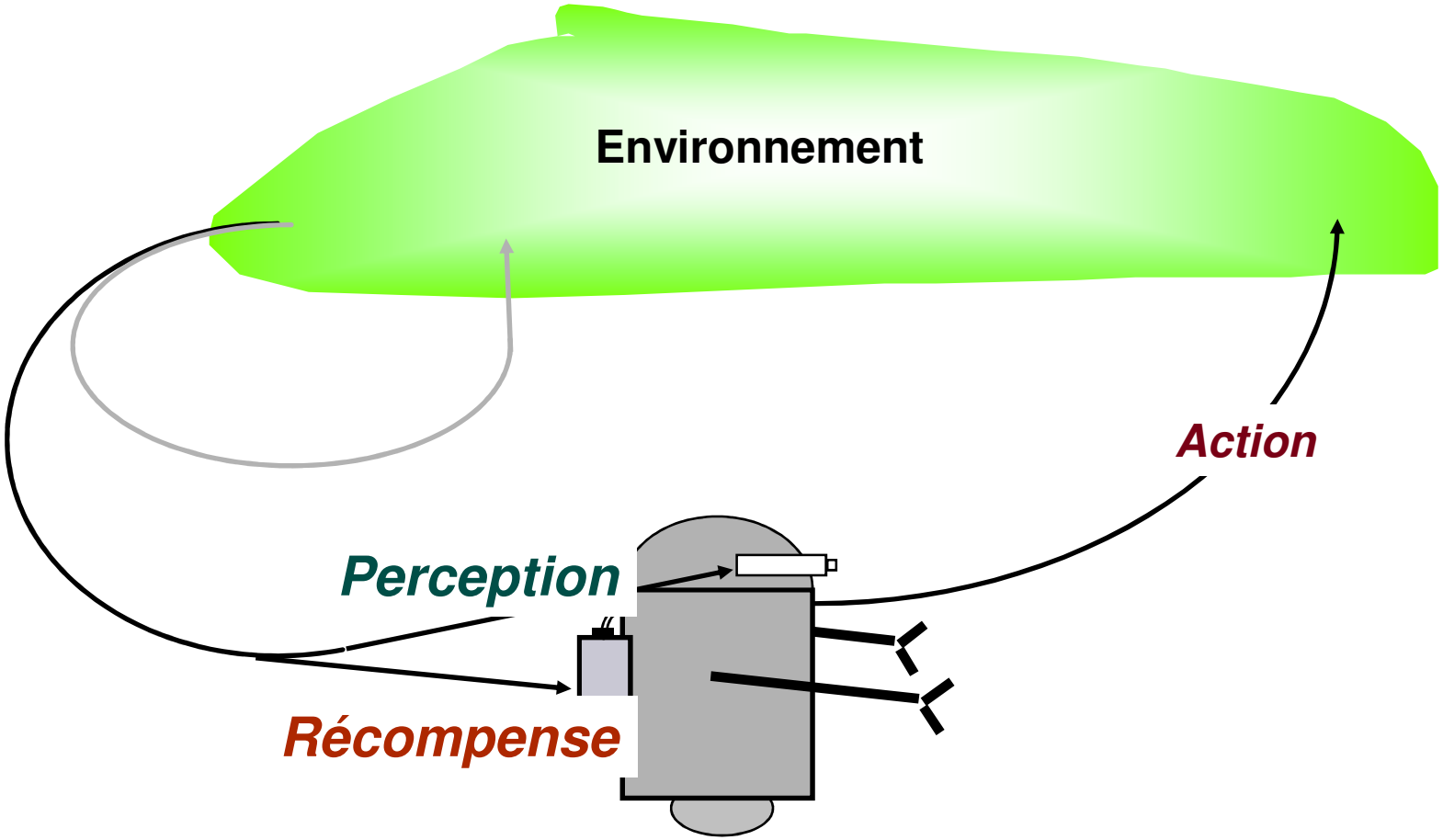
  Une animation



- Apprentissage
  - Essaie d'avoir une politique d'actions

    pour chaque situation
  - Très lié à l'IA : apprentissage par renforcement
  - Peut prendre plusieurs semaine de calculs pour l'apprentissage, mais peut être temps réel pour l'exploitation

Apprentissage par renforcement



Problème non uniquement lié à la production d'un geste

Apprentissage par renforcement

- *Temps* discret: t
- *États* : $s_t \in \mathtt{S}$
- *Actions* : $a_t \in \mathtt{A}(s_t)$
- *Récompenses* : $r_t \in \mathtt{R}(s_t)$
- L'agent : $s_t \rightarrow a_t$
- L'environnement : $(s_t, a_t) \rightarrow s_{t+1}, r_{t+1}$
- *Politique* : $\pi_t : \mathtt{S} \rightarrow \mathtt{A}$
  - Avec $\pi_t(s,a)$ = Prob que $a_t = a$ si $s_t = s$

T, R

- Les transitions et récompenses ne dépendent que de l'état et de l'action précédents : processus Markovien

Apprentissage par renforcement

- *Politique* :

  ensemble d'associations *situation* → *action* (une application)

  Une simple table ... un algorithme de recherche intensive

  Eventuellement stochastique

- *Fonction de renforcement* :
  - Définit implicitement le but poursuivi
  - Une fonction : (*état*, *action*) → *récompense* ∈ $\Re$
- *Fonction d'évaluation* *V(s) ou Q(s,a)* :
  - Récompense accumulée sur le long-terme
- *Modèle de l'environnement* :
  - Fonctions *T* et *R* : (*état(t)*, *action*) → (*état(t+1)*, *récompense*)

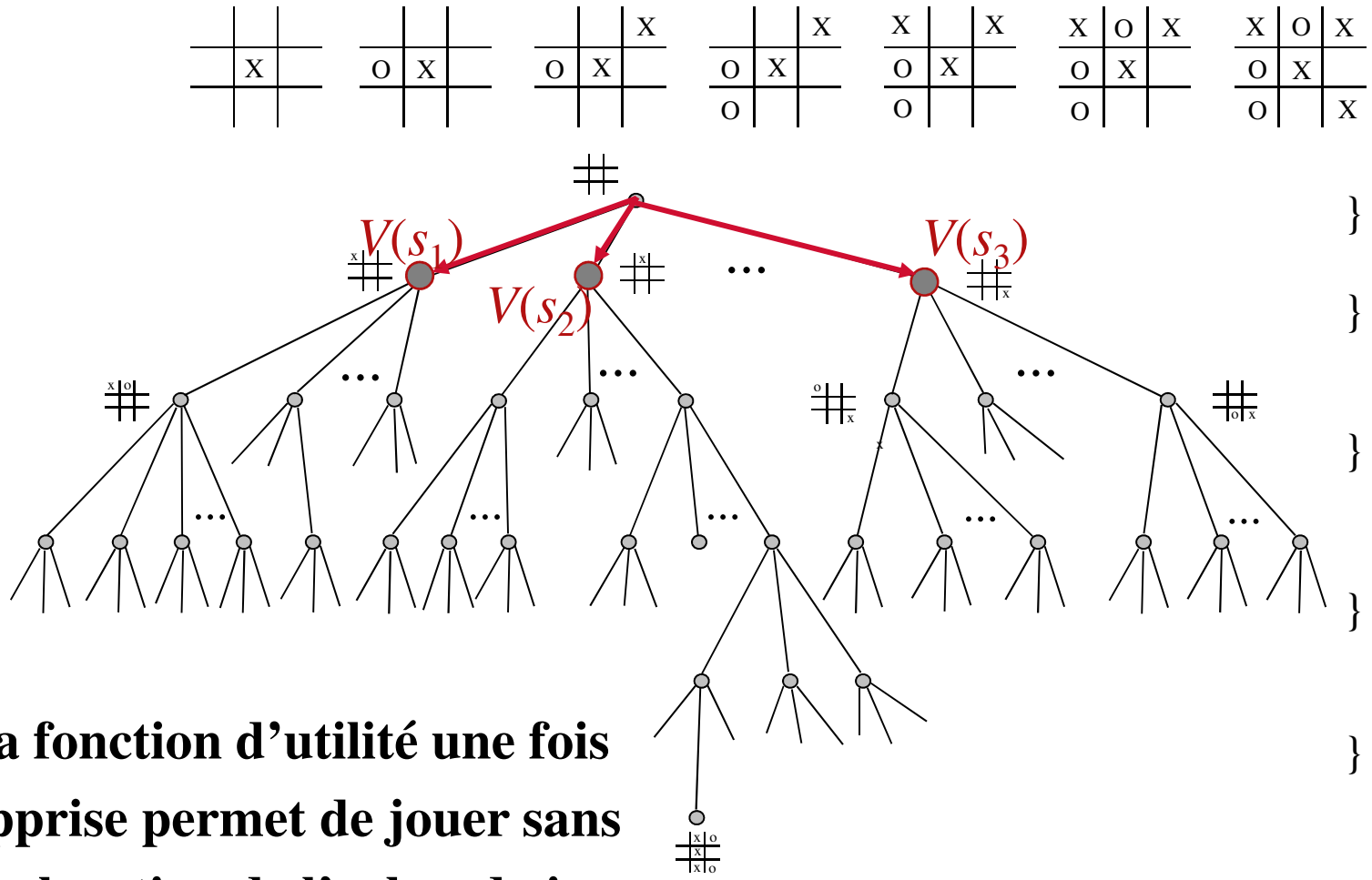
Apprentissage par renforcement

Principe :

- Choisir une action sans avoir besoin de faire une exploration (simulée) en avant

- Il faut donc disposer d'une fonction d'évaluation locale résumant une espérance de gain si l'on choisit cette action : *fonction d'utilité*
- Il faut apprendre cette fonction d'utilité : *apprentissage par renforcement*

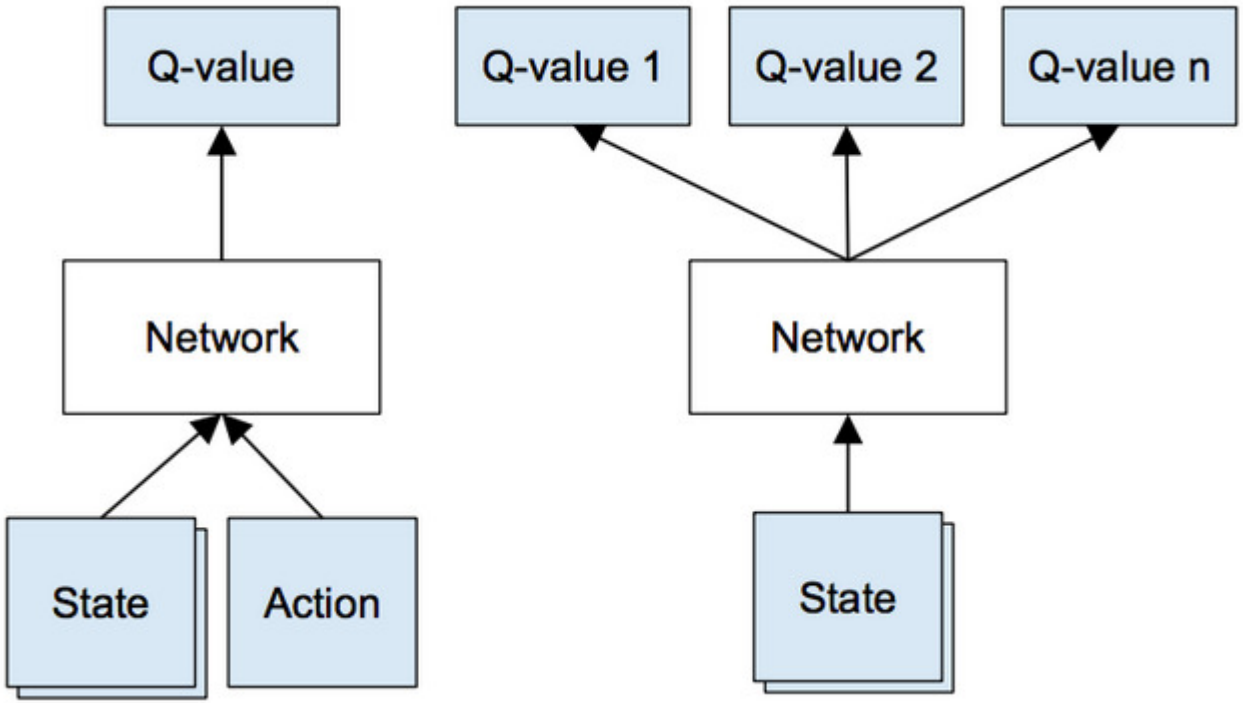
Apprentissage par renforcement



La fonction d'utilité une fois apprise permet de jouer sans exploration de l'arbre de jeu

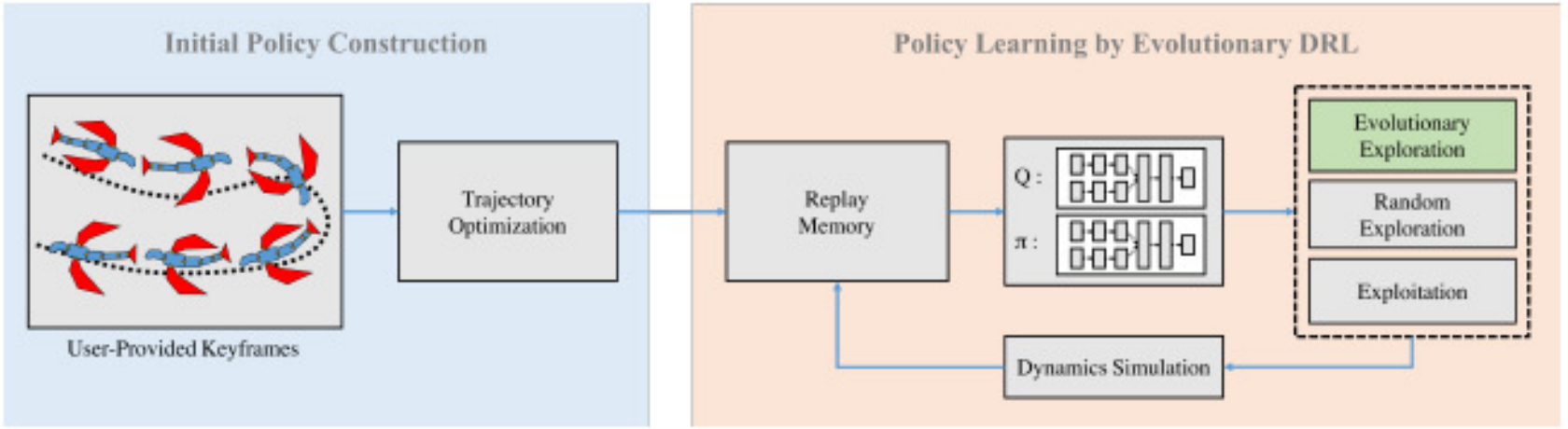
Deep Q Learning

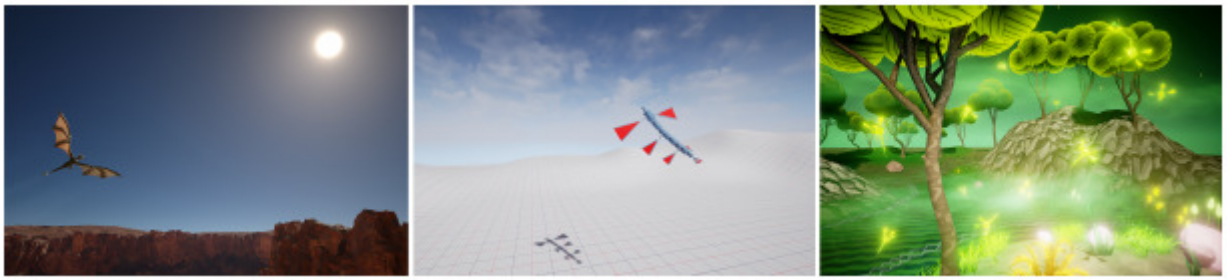
- Réseau de neurones bien adaptés au problème
  - Voir le cours Machine Learning & Animation



Optimisation / Apprentissage

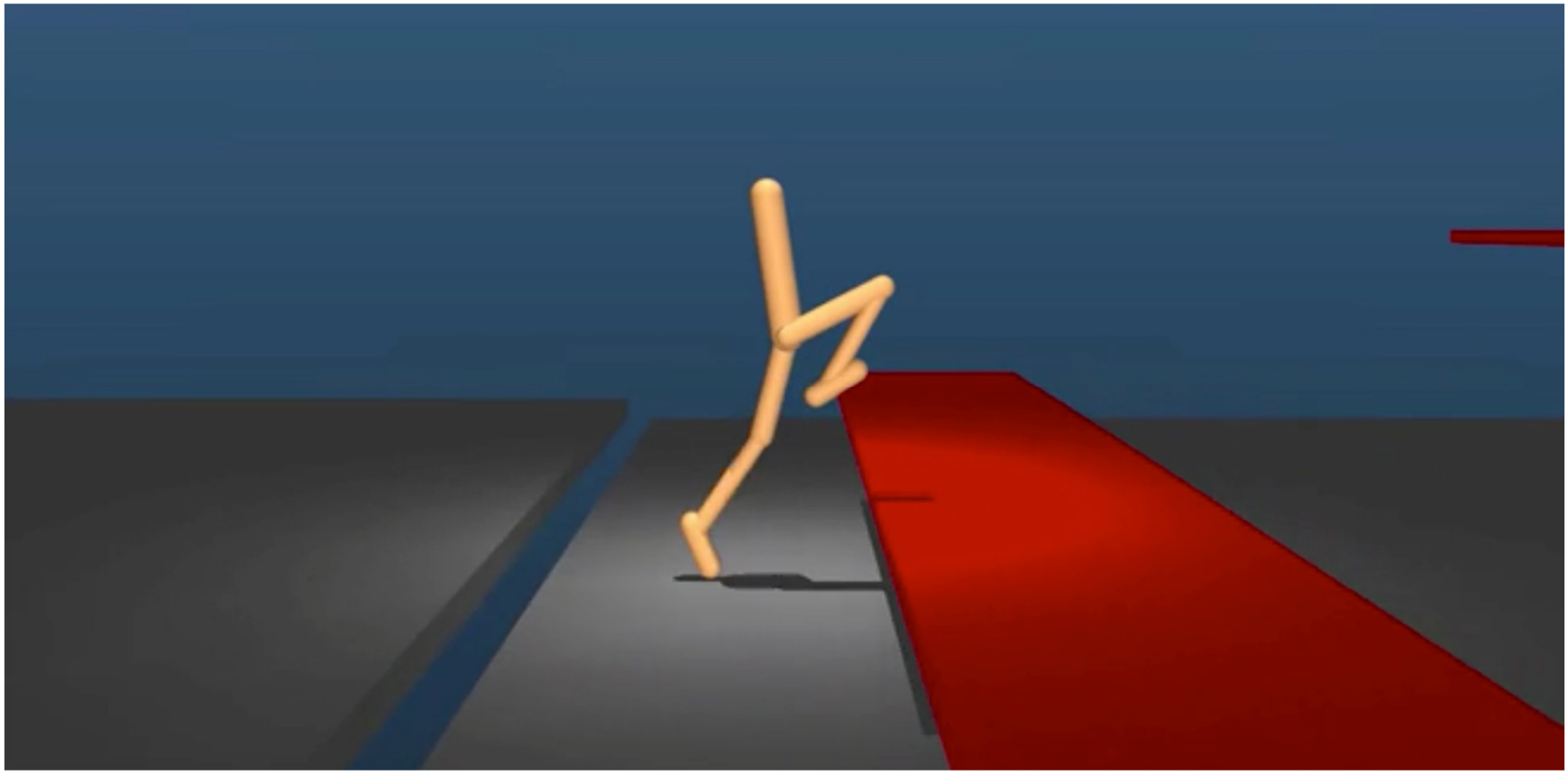
- Objectif : diminuer les temps de calculs
- Optimisation fournit la base de mouvement
  - Adapte la MoCap ou quand MoCap n'existe pas
  - Exemple : How to Train Your Dragon: Example-Guided Control of Flapping Flight





Learning to move

+vidéos



Conclusion : animation de perso