

HMIN 317 – Moteur de Jeux

Animation temps réel

Ce cours est largement inspiré des cours de **Rémi Ronfard**, ***Marc Moulis et Benoit Lange***.

Chemins et trajectoires

Animation de modèles

Animation réaliste

Animation faciale

Animation de caméras

Chemins et trajectoires

La manière la plus simple de représenter un chemin de déplacement en 3D :

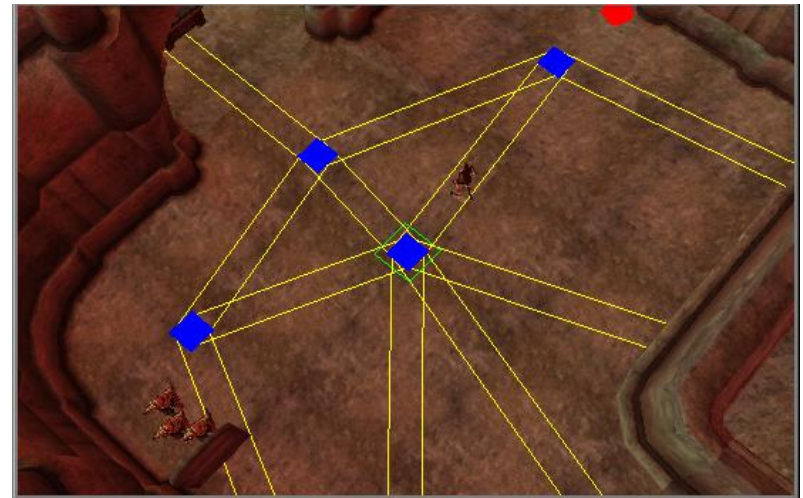
- définir une série de points (ou nœuds) dans l'espace,
 - reliés 2 à 2 par des segments.
- ligne brisée, représentant le chemin.

Selon les besoins, les segments peuvent être orientés, et ainsi définir un point de départ et un point d'arrivée.

Lorsque un nœud est connecté à plus de 2 voisins, on obtient un graphe de déplacement.



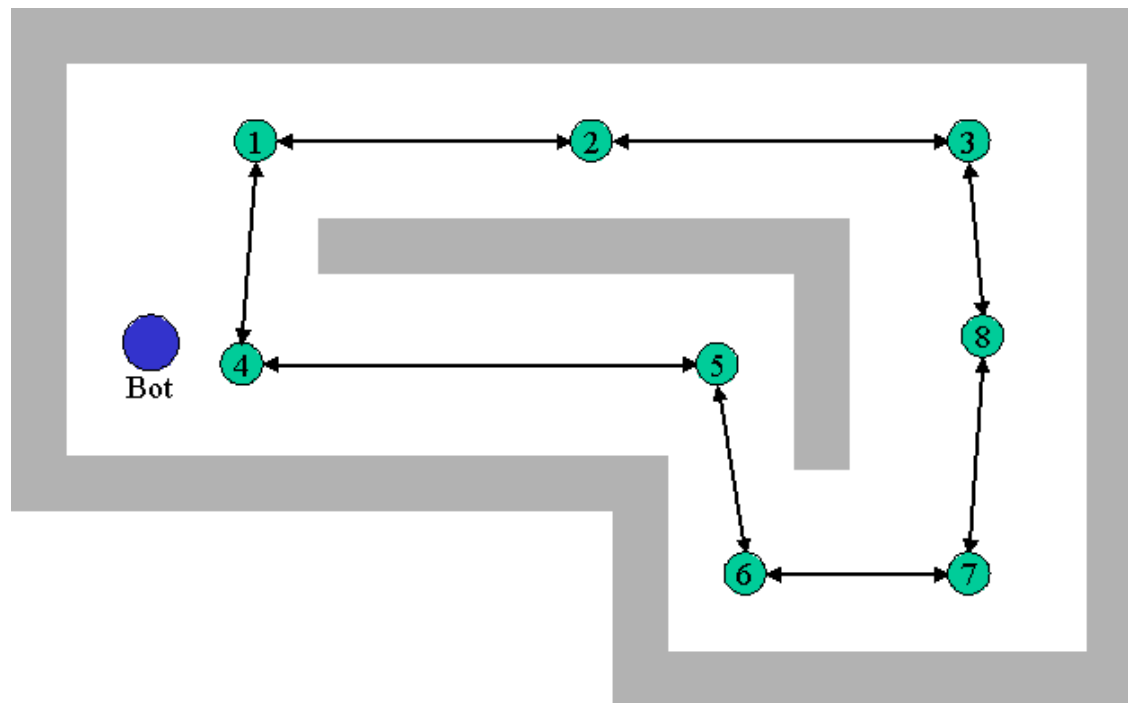
Ligne brisée représentant un chemin



Exemple de graphe de déplacement :
chaque nœud est connecté à un ou
plusieurs voisins

Ce type de représentation pour la **recherche de chemin** :

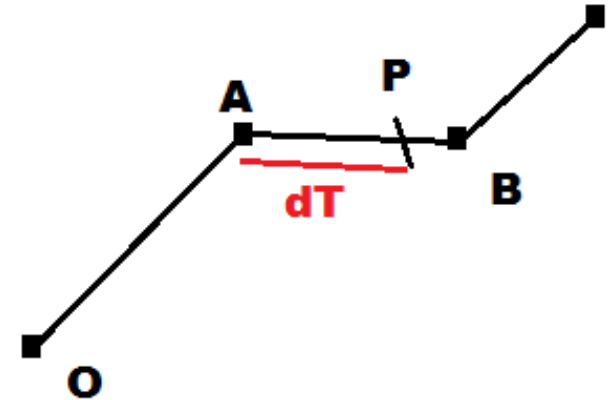
- exploration de proche en proche tous les nœuds d'un graphe,
- déterminer une trajectoire à suivre dans l'environnement (waypoint).



Exemple de chemin placé dans un environnement. Les nœuds du chemin servent de repères à l'entité lors de la planification de ses déplacements.

Lignes polygonales

- Problème :
 - trajectoire associée est souvent **non-naturelle**, avec des changements de direction brutaux au niveau des nœuds.
- Raffiner énormément la ligne
 - grand nombre de nœuds et ainsi l'adoucir,
 - travail d'édition fastidieux et le stockage mémoire important.



→ Splines : plus adaptée de
génération d'un chemin courbe

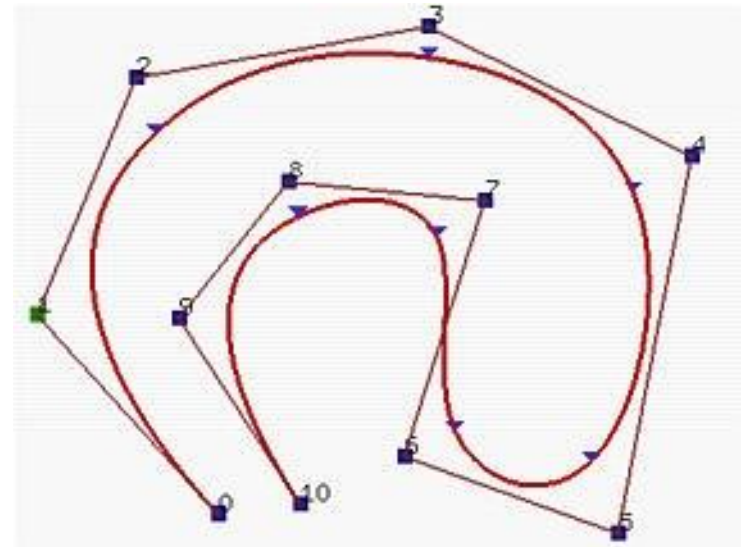
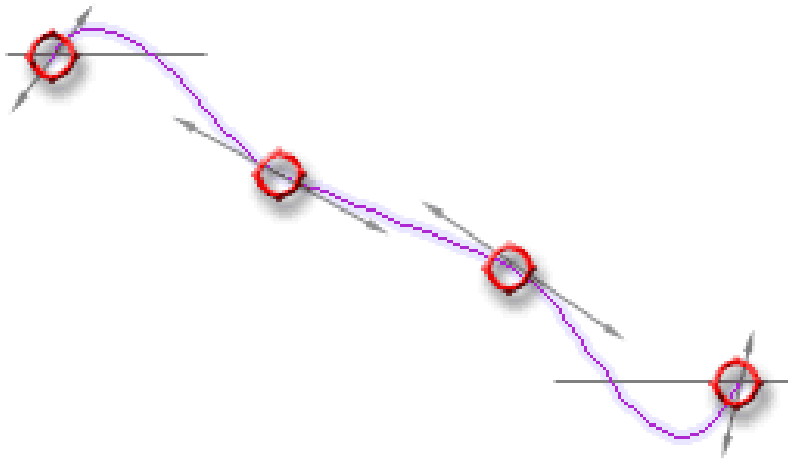
Splines

- **Fonction définie par morceaux**, chaque morceau étant décrit par un polynôme.
- Représentation d'une courbe :
 - spécification d'une série de **points de contrôle** à des intervalles réguliers,
 - fonction permettant de calculer tous les points entre les points de contrôle.
- Il existe plusieurs types de splines, avec des propriétés différentes:
 - Courbes de Bézier, B-Splines, Catmull-Rom, Cardinal, Hermite
 - NURBS (Non-Uniform Rational B-Splines)

On différenciera tout particulièrement 2 catégories de splines:

- **Splines interpolantes** (la courbe passe par les points de contrôle)
- **Splines approximantes** (la courbe ne passe pas par les points de contrôle)

Remarque: certaines splines sont quasi-interpolantes (la courbe passe par certains des points de contrôle)



Exemple de spline interpolante (gauche) et de spline approximante (droite).

Points de contrôle et degrés

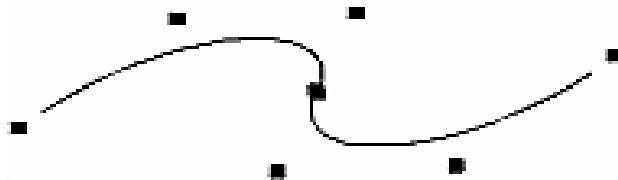
Le degré d'une courbe détermine le nombre de points de contrôle nécessaires pour définir la courbe:

- degré 2 (quadratique) : 3 points de contrôle
- degré 3 (cubique) : 4 points de contrôle.

On peut construire une courbe plus longue de même ordre en rattachant des morceaux de courbe.



A Cubic Curve



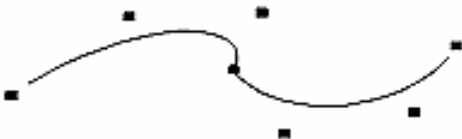
A Piecewise Cubic Curve

Le degré d'une courbe affecte également le comportement de la courbe lorsqu'un des points de contrôle est déplacé. En général, les courbes de trop haut degré sont sujettes à des effets oscillatoires indésirables.

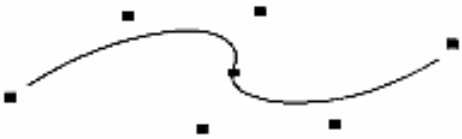
Continuité d'une courbe



No Continuity



**C0 Continuity
(positional)**



**C1 Continuity
(tangential)**



**C2 Continuity
(curvature)**

On parle de continuité C_n d'une courbe pour exprimer le fait que les n premières dérivées des morceaux de courbe consécutifs sont égales au niveau des points de jonction entre les morceaux.

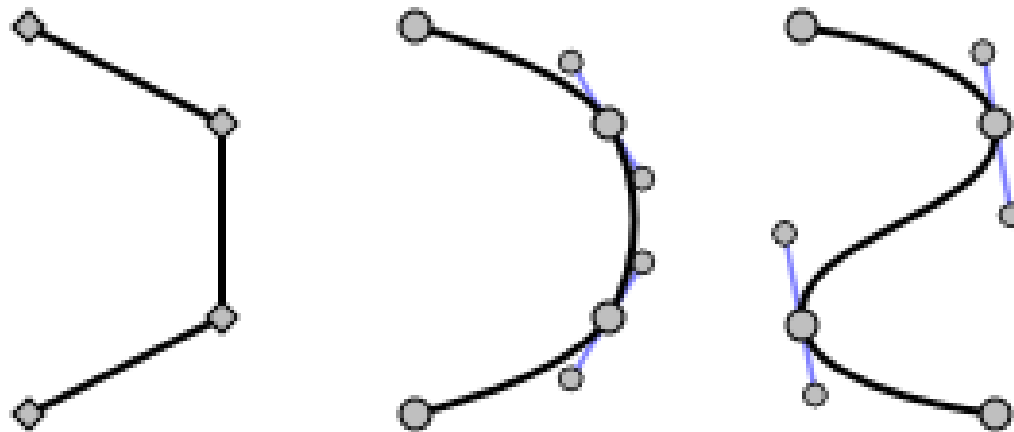
Continuité C_0 : la courbe est continue uniquement pour les positions

Continuité C_1 : les tangentes à la courbe (dérivées premières) sont constantes au niveau des points de jonction

Continuité C_2 : la courbure des courbes (dérivées secondes) est constante au niveau des points de jonction

Le principe d'édition d'une spline est d'associer à chacun des points de contrôle une **tangente à la courbe**, qui déterminera donc de quelle manière la courbe doit s'orienter dans le voisinage local au point de contrôle.

Techniquement, la tangente à un point de contrôle est souvent calculée en utilisant la position des points de contrôle précédent et suivant.



Génération d'une spline (interpolante) à partir d'un maillage de points de contrôle et des tangentes associées.

Splines de Bézier quadratiques

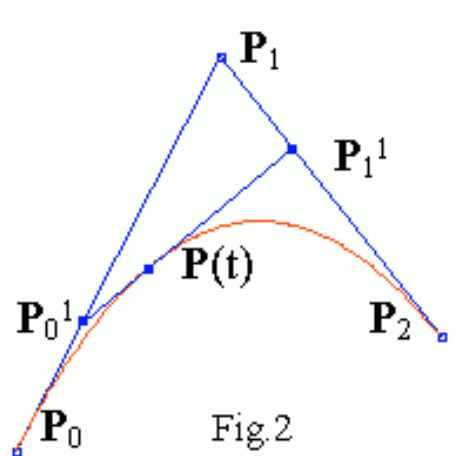


Fig.2

Bézier quadratique

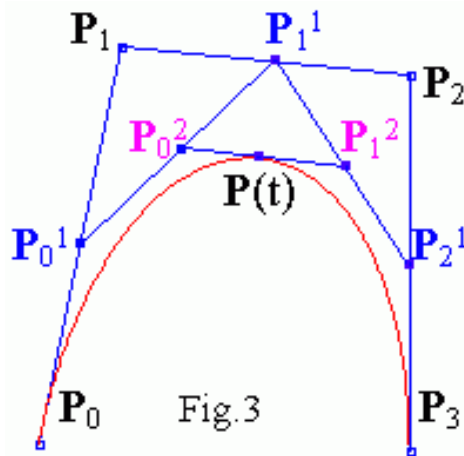


Fig.3

Bézier cubique

On les obtient par l'algorithme de De Casteljau, en réalisant une interpolation linéaire des interpolations linéaires entre les points de contrôle P_0 , P_1 et P_2 .

$$P_{01} = (1-t)P_0 + tP_1$$

$$P_{11} = (1-t)P_1 + tP_2$$

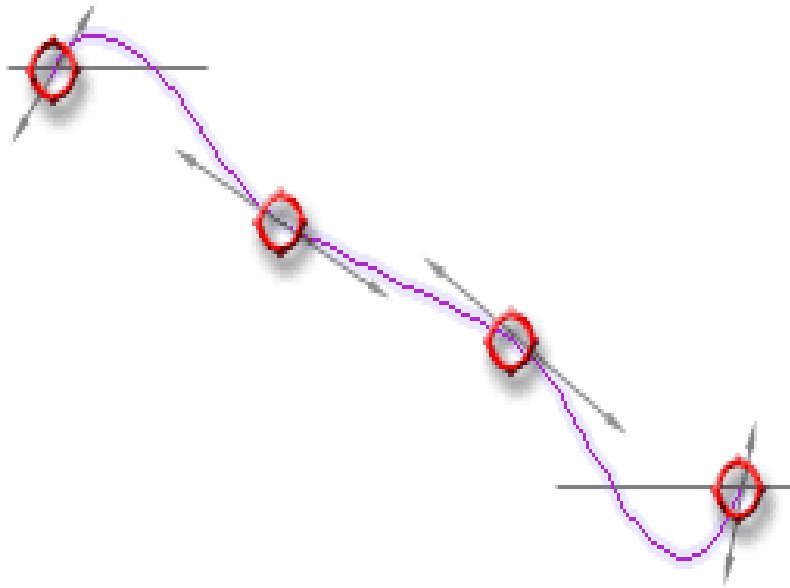
$$P(t) = (1-t)P_{01} + tP_{11}$$

Splines de Bézier cubiques

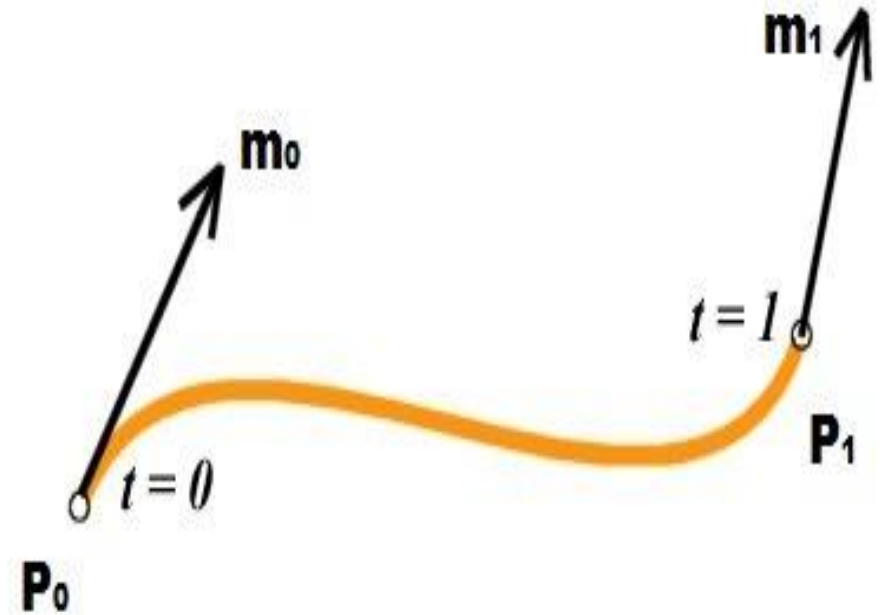
De la même manière, il est possible d'interpoler une courbe entre $(n+1)$ points de contrôle. Si $n=3$, on obtient une spline de Bézier cubique.

Les splines de Bézier ont une continuité $C1$.

Splines de Catmull-Rom et Hermite

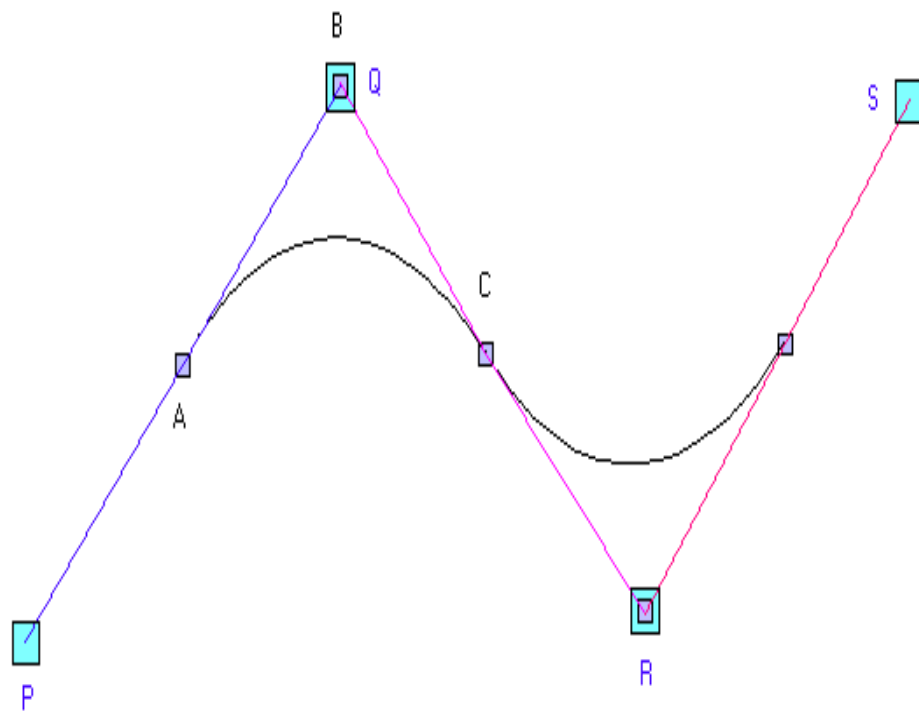


Catmull-Rom

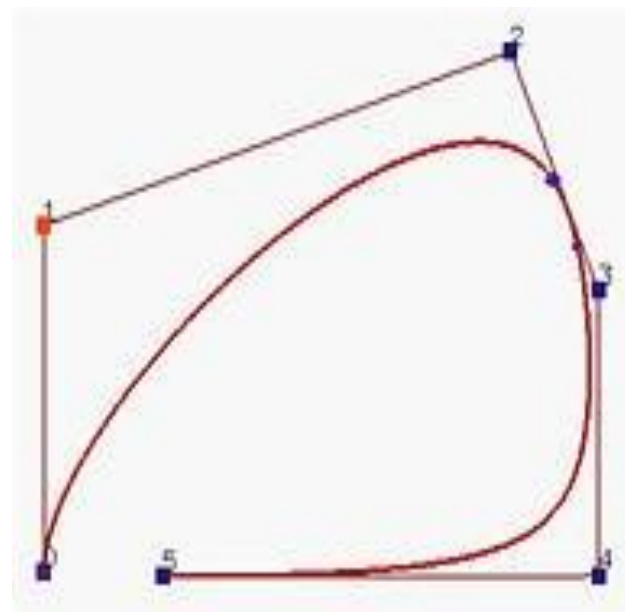


Hermite

B-Splines et NURBS



B-spline

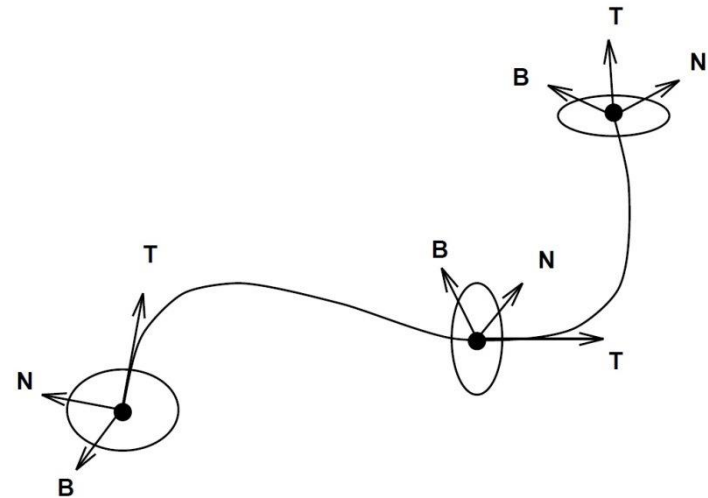
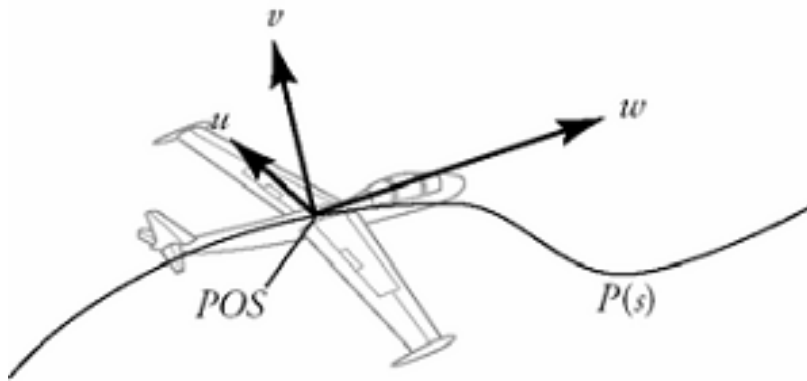


NURBS

Repère de Frenet

Une fois le chemin déterminé, il faut être capable d'orienter l'objet qui va se déplacer le long de ce chemin.

Une méthode permettant de déterminer automatiquement une telle orientation est le calcul du **repère de Frenet**.



Visualisation du repère de Frenet le long de la courbe

Calcul du repère de Frenet

$$\begin{aligned}\vec{T}(s) &= \frac{\vec{x}'(s)}{\|\vec{x}'(s)\|} \\ \vec{B}(s) &= \frac{\vec{x}'(s) \times \vec{x}''(s)}{\|\vec{x}'(s) \times \vec{x}''(s)\|} \\ \vec{N}(s) &= \vec{B}(s) \times \vec{T}(s) .\end{aligned}$$

En pratique, on calcule rarement le véritable repère de Frenet :

- orientation globale de l'un des axes B ou N est connu,
- simplification des calculs (on évite le calcul de la dérivée seconde).

Exemples typiques : caméra en mode « poursuite », connaissance de l'orientation de la « verticale » de l'objet, etc...

1. Déterminer l'axe T :

- à partir du vecteur déplacement le long de la courbe (= au repère de Frenet),
- ou en orientant vers une position cible,

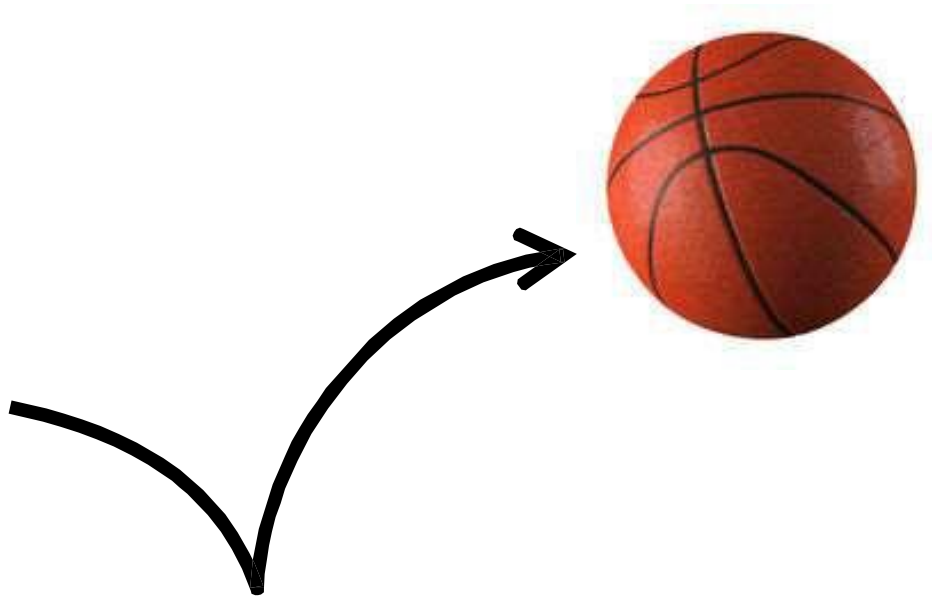
2. Fixer l'orientation d'un des deux autres axes (eg B dans le sens de la verticale du monde),

3. Déduit N par simple produit vectoriel entre B et T.

Types d'animation dans les jeux

1. D'objets rigides

- animer des transformations

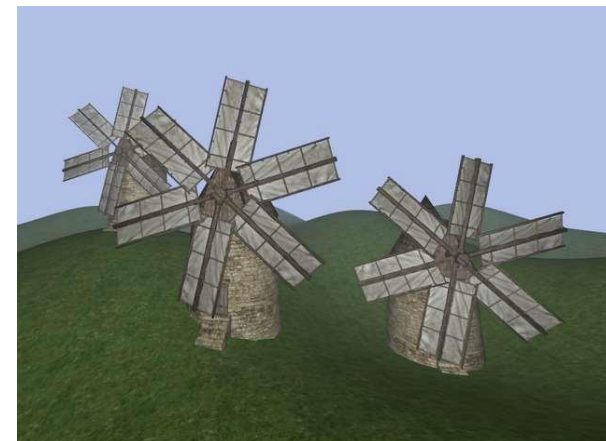


(6 DoF par objet)

Types d'animation dans les jeux

1. D'objets rigides

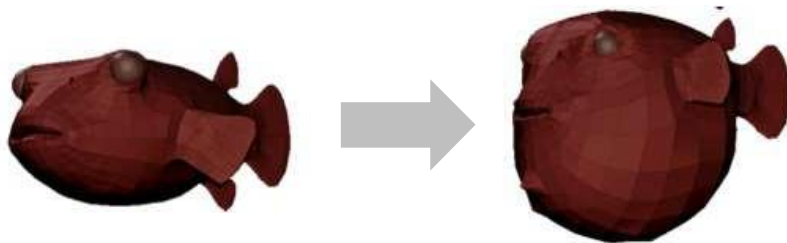
- Ou d'objet fait



Types d'animation dans les jeux

2. Déformation de formes libre (Free-Form)

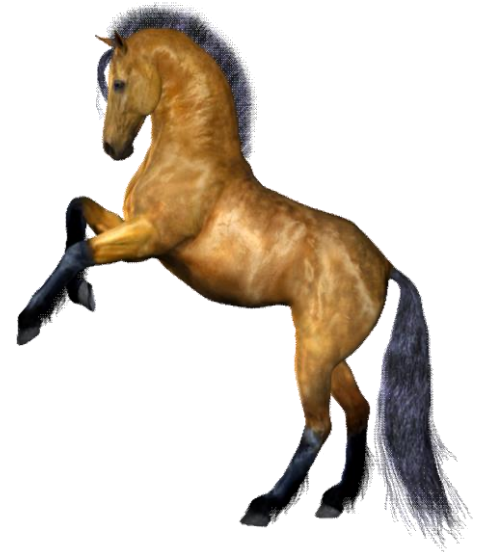
- Transformation générique de l'objet



Types d'animation dans les jeux

2. De modèles articulés

- Squelette
- La plupart des personnages virtuels



DoF par keyframe

Rigid



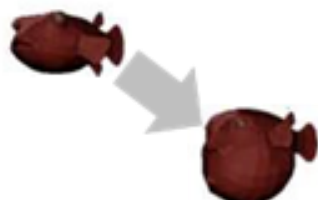
6 DoF per object
(or, e.g., 9, with anisotropic scaling)

Articulated



~50-100 DoF per object
(e.g. 3 DoF per joint x 25 joints)

Free form



300-10.000 DoF per object
(e.g. 3 per-vertex)

Animation dans les jeux

Non procedural

- **Assets!**
- **Control:** easy.
full control by artists
(e.g. for dramatic fx)
- **Realism:** hard
it's up to the artist skill
- **Flexibility:** little
Doesn't adapt to env.
- (consumes RAM)

Procedural

- **Physic engine**
- **Control:** hard
- **Realism:** easy
built-in physical laws
- **Flexibility:** great
Adapts to env / contexts
- (consumes GPU)



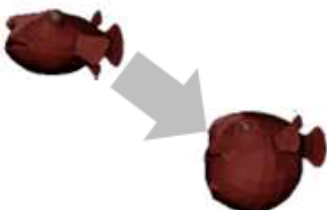
Animation dans les jeux

- Ou... un mélange
- Exemples
 - animations primaires: scriptées
 - animations secondaires: calculées
 - personnages vivants: scripté
 - caractères morts: calculés (ragdolls)
- ...
- Une frontière entre CG et de techniques interactives

Animations scriptées

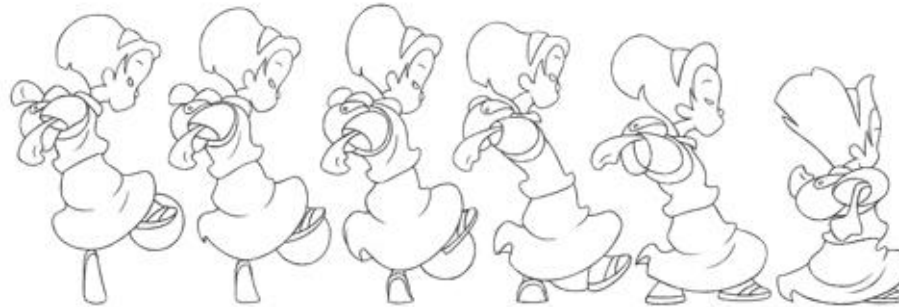
- D'objets en sous-parties rigides
 - joints compris: robots, voitures...
 - → “animations cinématiques directe” (changements dynamique des transformations)
- D'objets articulés déformables
 - Avec un squelette interne
 - par exemple: la plupart des personnages virtuels :
 - humains / animaux / monstres / n'importe quoi entre
 - “skinning” / “rigging”
- D'objets génériques déformables
 - par exemple des visages, un parapluie, un tissu à membrane...
 - «Animations par sommet» / «fusion de formes/blend shape» / «cibles de morphing»

Objets solides 3D

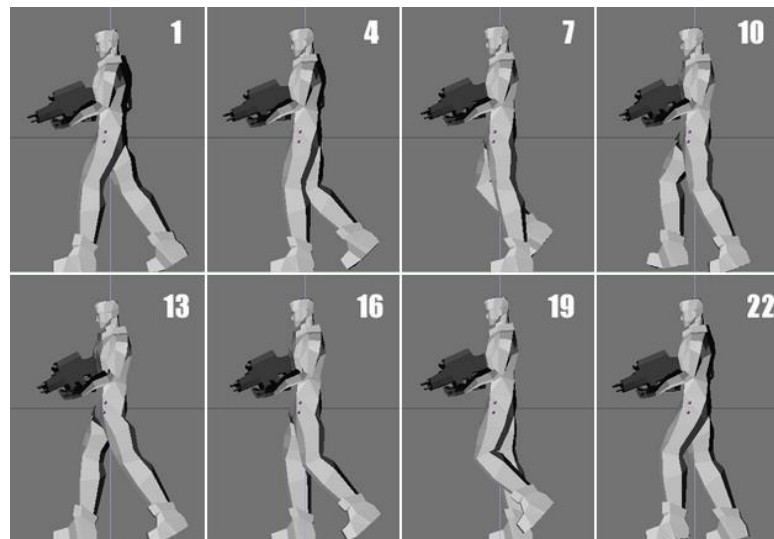
	<div> <div>Non Procedural</div> <div>(ASSETS)</div> </div>	<div> <div>Procedural</div> <div>(PHYSIC ENGINE / ETC)</div> </div>
Rigid 	<div>Pre-made transforms</div>	<div>Rigid body dynamics</div>
Articulated 	<div>Skeletal Animations</div>	<div>Ragdolling</div> <div>Inverse kinematics</div>
Free form 	<div>Blend-Shapes</div>	<div> (generic) deformable object simulation <i>usually too expensive</i> </div> <div>Cloth/garments</div> <div>Ropes</div>

Animation de modèles : vertex tweening

La méthode d'animation la plus basique s'inspire directement des méthodes d'animation 2D traditionnelles.

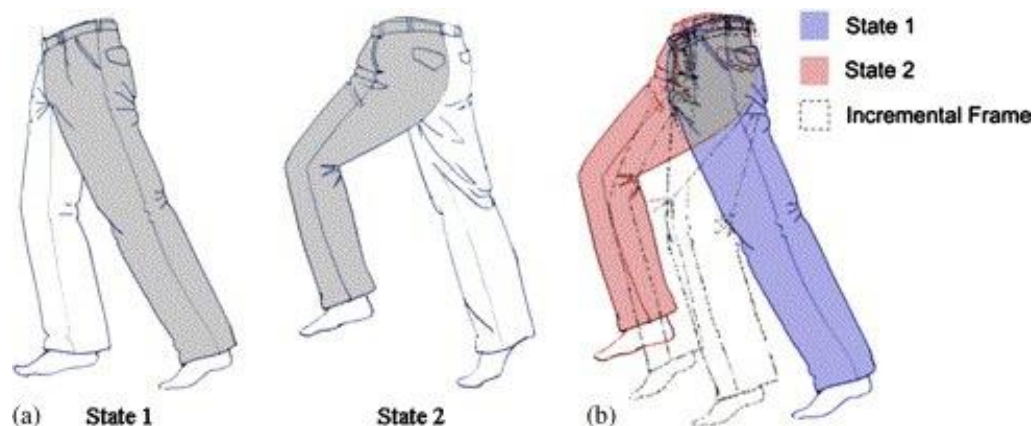


Pour chacune des images de l'animation, on crée un maillage 3D qui représente l'étape de l'animation.



Vertex tweening

Pour produire l'animation, les positions des vertices de la géométrie sont linéairement interpolés 2 à 2 (vertex tweening).



Technique simple à mettre en œuvre, mais présente de nombreux défauts:

- **maillage complet pour chacune des poses** de l'animation (coûteux en mémoire)
- Si les étapes de l'animation sont trop éloignées les unes des autres : déformations visibles du maillage (**interpolation linéaire** des positions des vertices)
- maillages d'une animation doivent être **strictement identiques** (topologie/géométrie)
- même si différents modèles géométriques ont la même animation, **duplication et animation séparée**

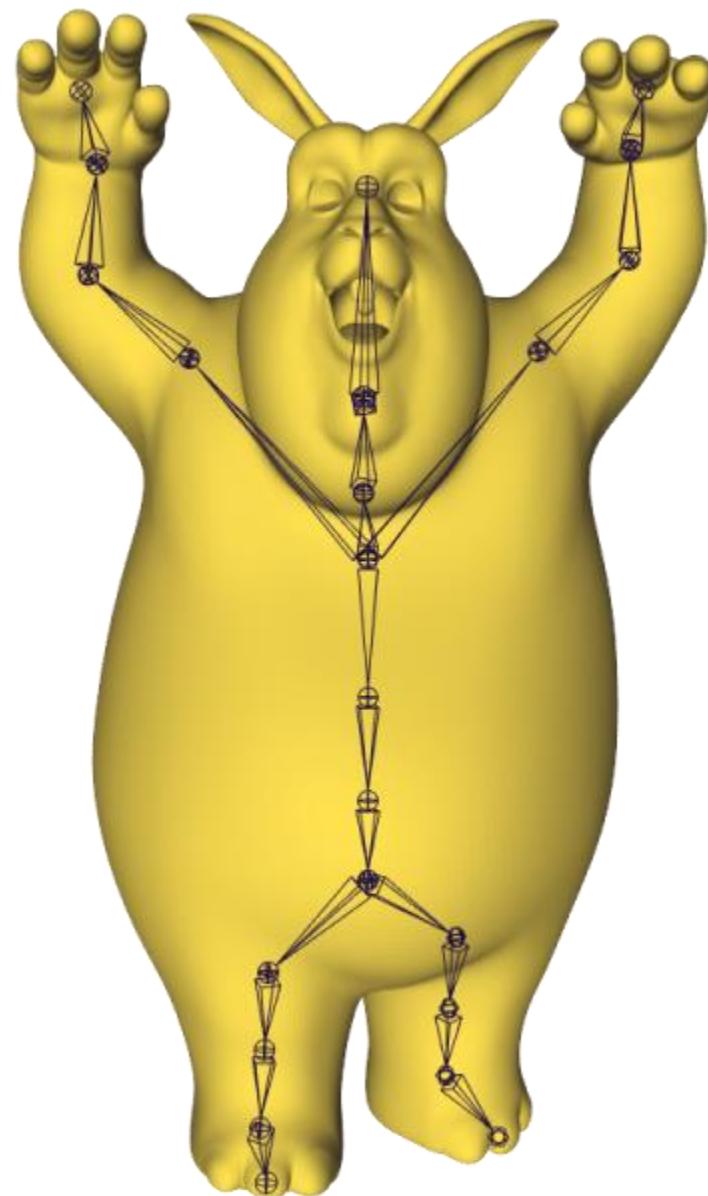
Rigging et skinning

- Alternative au vertex tweening :
 - structure de données permettant de **séparer les données d'animation du maillage 3D animé.**

Animation appliquée sur une hiérarchie de transformations :

- **squelette** de l'objet à animer

Un modèle géométrique séparé, la skin, servira d'enveloppe au squelette et sera déformé par celui-ci



Rigging & skinning

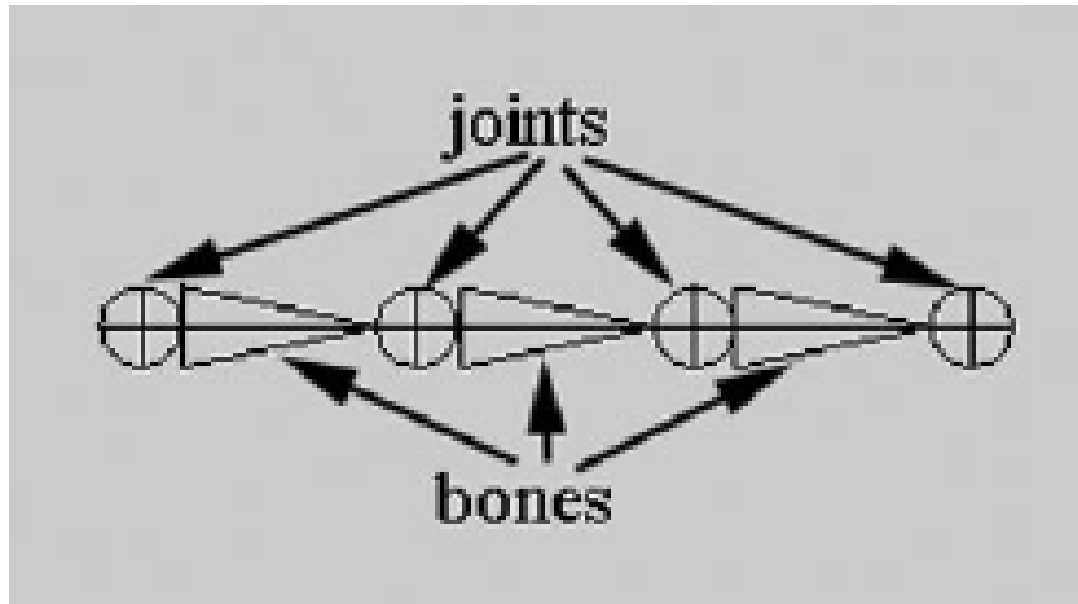
Squelette d'animation composé d'une **hiérarchie de transformations** (matrices 4x4) :

- Chaque nœud de la hiérarchie représente une **articulation**.
- Le squelette d'animation est également appelé « **rig** ».
- Sa racine est habituellement située au niveau des hanches.



Armatures

Chaque nœud (ou paire de nœuds) de transformation de la hiérarchie est traditionnellement appelé « bone », par analogie avec les os constituant un squelette.



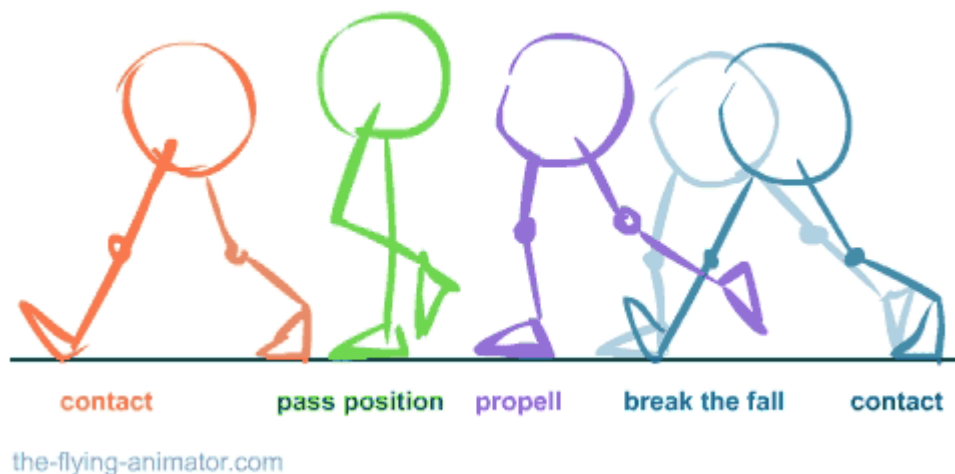
Deux représentations possibles pour les bones:

- Un bone est représenté par la liaison entre deux joints (nœuds de la hiérarchie)
- Un bone est associé à un nœud de transformation de la hiérarchie (le bone et le joint sont indifférenciés)

Animation par keyframes

Les animations du squelette sont stockées sous la forme de **keyframes**, série de poses pour **chacun des nœuds du squelette** :

- Vecteur 3D pour le stockage de la translation
- Quaternion pour le stockage de la rotation



Le format d'animation résultant est donc très compact (7 floats stockent l'équivalent d'une matrice 4x3 rotation/translation).

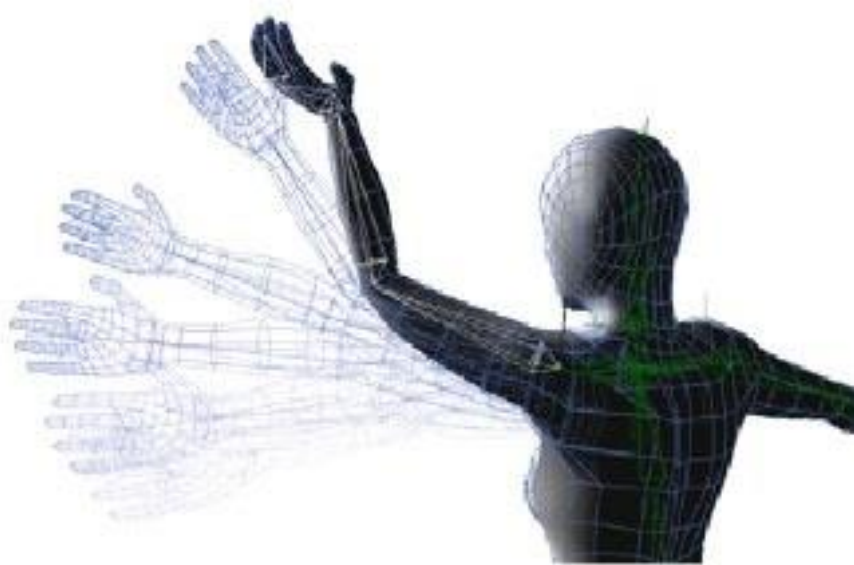
Les keyframes sont pré-calculées à partir:

- des données produites par les animateurs
- des données issues de la motion capture

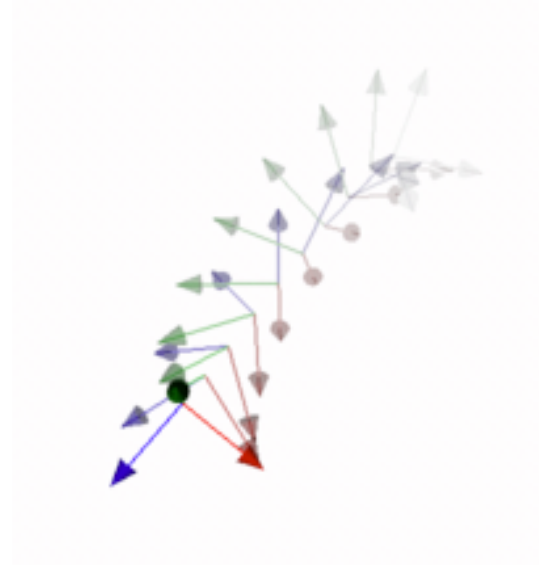
Interpolation

L'animation entre les différentes keyframes se fait par interpolation

- Généralement **linéaire** (LERP) pour la translation
- **Sphérique** pour l'orientation (SLERP sur les quaternions)



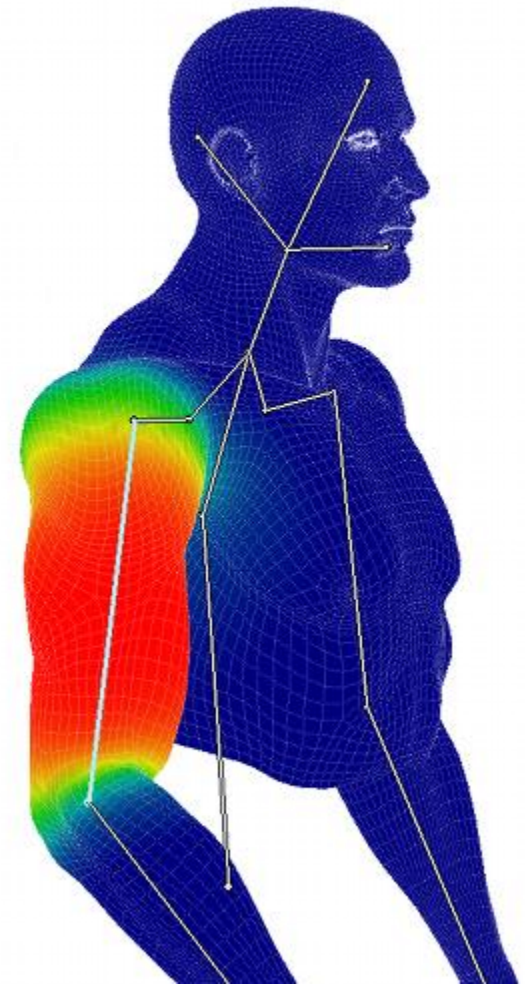
Animation du squelette par interpolation
entre les poses



Interpolation sphérique
d'un quaternion

Skinning

- Squelette d'animation construit :
 - lui associer un ou plusieurs modèles géométriques : **enveloppe**,
 - enveloppe déformée lors de l'animation.
- Historiquement, trois techniques :
 - Rigid bones
 - Hard skinning
 - Soft skinning



Rigid bones

Enveloppes du type « rigid bones » :

- modèles les plus simples,
- Associer à chacun des bones du squelette, un modèle géométrique différent.
- Chaque nœud de transformation du squelette associé à **un seul** maillage indépendant.

Seul la transformation du bone dans l'espace est appliquée sur les sommets des maillages → **très rapide**

Transformation réalisée par le hardware (pipeline fixe)
→ c'était la méthode d'animation privilégiée avant les pipelines programmables.

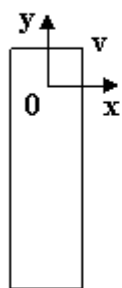


Modèle « rigid bones ».

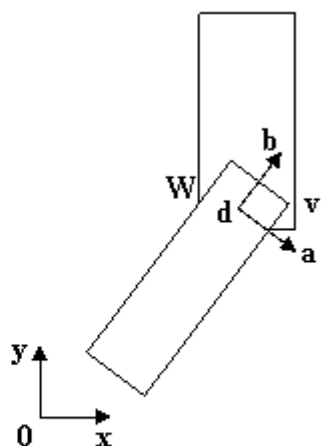
Artefacts

Artefacts visuels dans le modèle lors de l'animation :

- Interpénétration des différents bones
- Apparition d'espaces disjoints (cracks) entre les bones

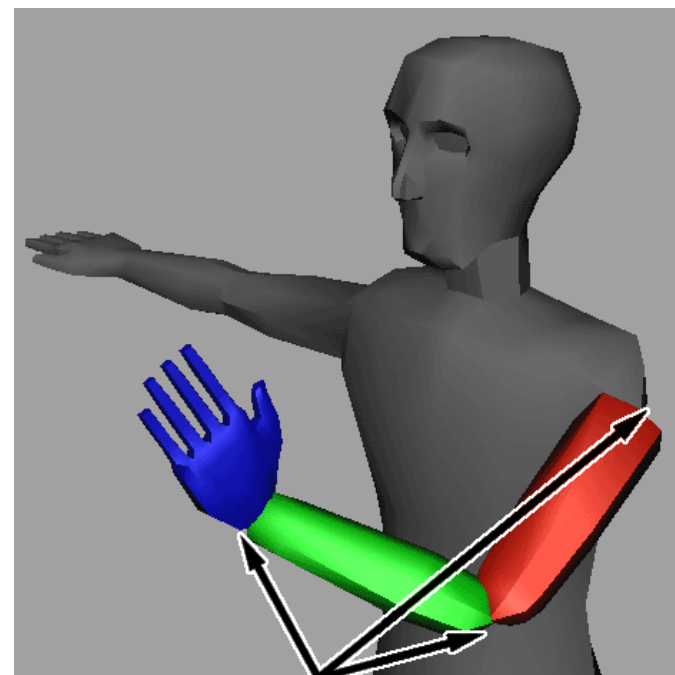


Vertex v defined in joint's local space



Vertex v transformed to world space using matrix W

Animation de type « rigid bones ». Des artefacts apparaissent au niveau des jointures.



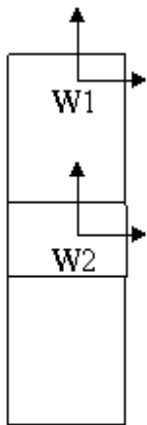
Unnatural cracks at the joints

Skinning

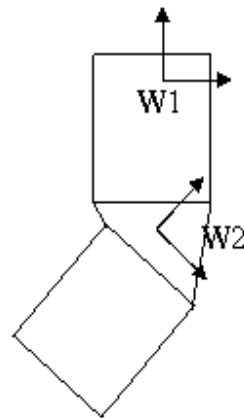
Concept de « skin » : un seul maillage pour l'ensemble de l'enveloppe.

A chacun des vertex de la skin : **index du bone** dans le squelette utilisé pour la transformation lors de l'animation (hard skinning).

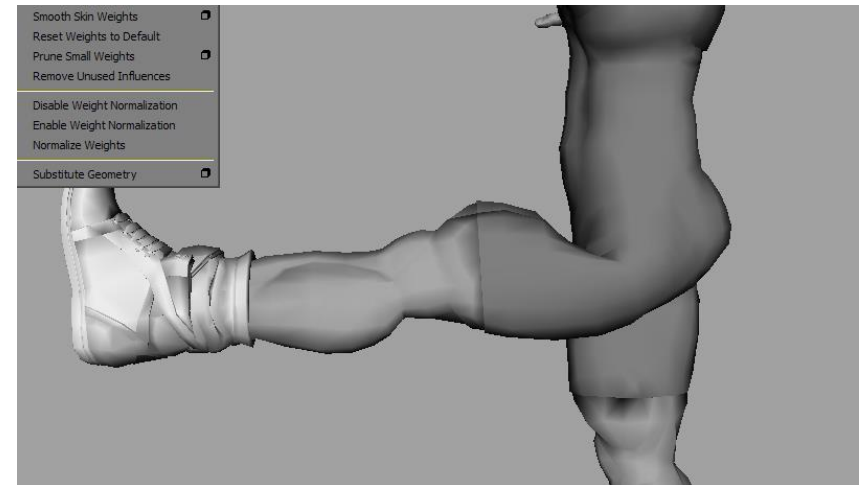
Certains défauts notables sont toujours présents : sommets ne sont attachés à **un seul** bone.



An unbent knee with skin attached to joints 1 and 2



Every vertex is attached to exactly one joint, so as the knee bends, we get some distortion

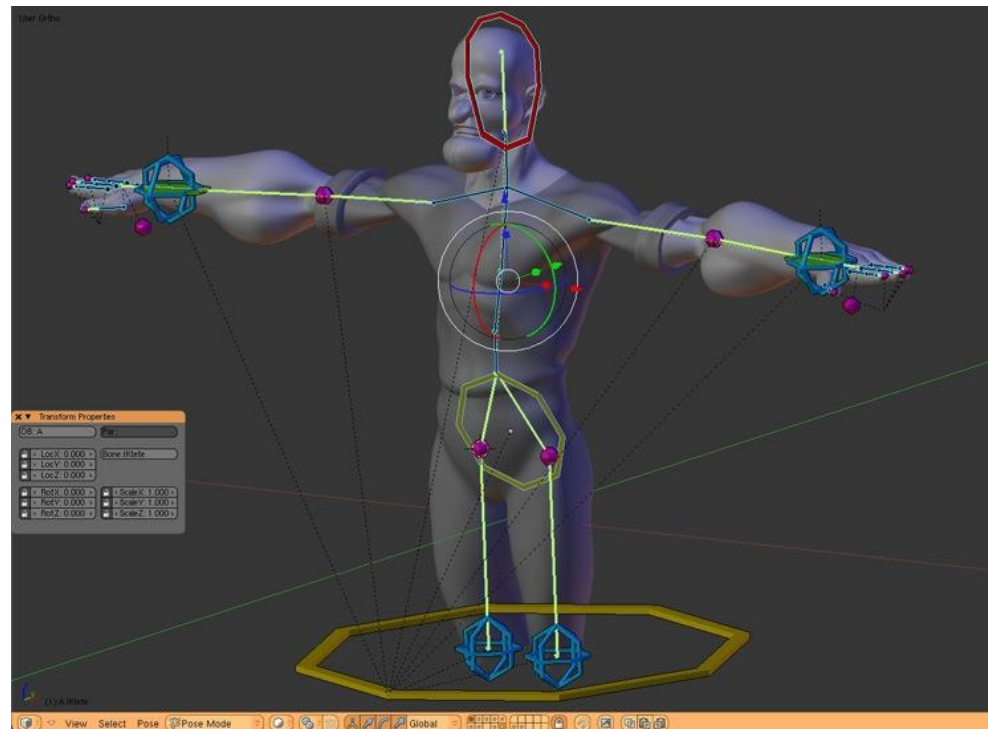


Déformation à la jointure de la cuisse avec la hanche.

Skinning

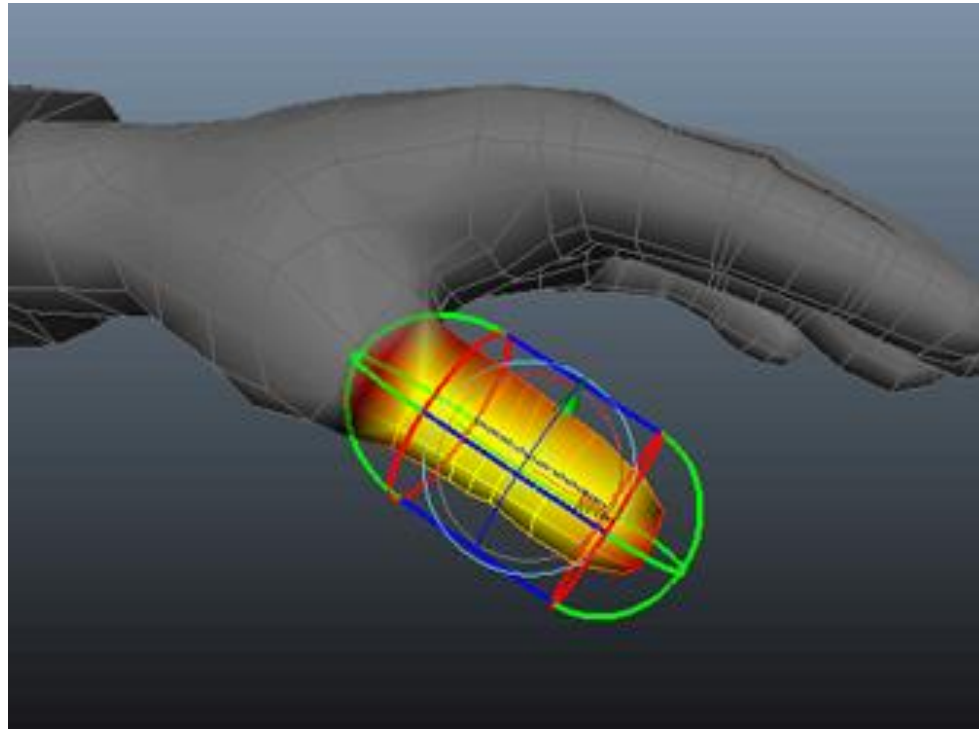
Placer sa skin dans un état initial connu par l'animateur :

- pour un humanoïde, généralement debout avec les bras à l'horizontale : « T-pose »
- initialisation des matrices de transformation du squelette dans le **même espace que celui de la skin** (skin binding).



Skin blending

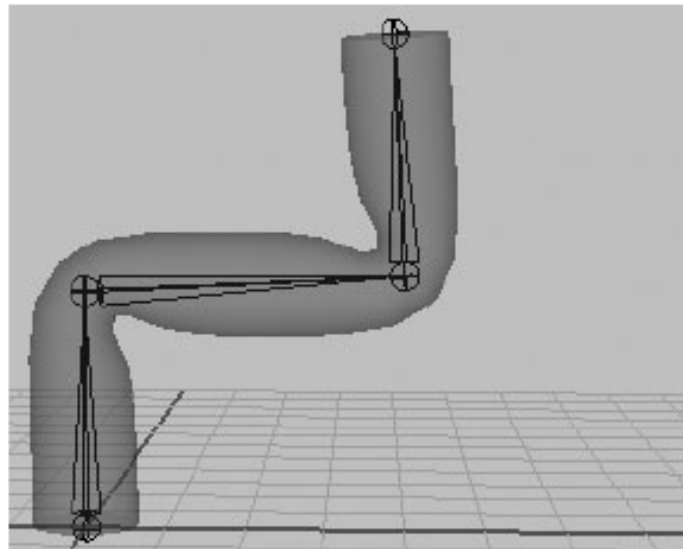
Puis, à l'aide d'outils spécifiques, l'infographiste va peindre directement sur le modèle les influences des différents bones du squelette.



Affectation des poids de skinning.

Smooth skinning

- On fournit au moteur de rendu toutes les **matrices de transformations**
- stocker pour chaque vertex de la **skin les indices des matrices et les poids de skinning** (également appelés Beta-weights)
 - influences des bones combinées pour chacun des vertex de la skin dans le **vertex shader**



Déformation d'un maillage par smooth skinning

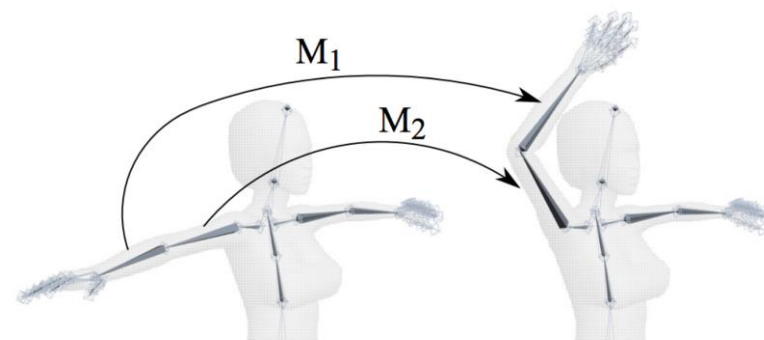
Linear blending (smooth skinning)

Chaque matrice de skinning est de la forme

$$M_j = T_j \cdot B_j^{-1}$$

avec

- M_j : Matrice de skinning de la palette
- T_j : Matrice du bone (animé) dans le repère global centré
- B_j^{-1} : Inverse de la matrice de pose du bone



L'application des poids de skinning se fait par moyenne pondérée des positions du vertex obtenues après transformation par les matrices de skinning associées.

$$\mathbf{p}_i' = \sum_j w_{ij} \cdot M_j \cdot \mathbf{p}_i$$

avec

- \mathbf{p}_i' : Position du vertex après skinning
- w_{ij} : Influence du jème bone du vertex
- M_j : Index de la jème matrice de skinning
- \mathbf{p}_i : Position originale du vertex dans la skin

Rappel : les poids de skinning sont normalisés (leur somme vaut 1).

Vertex tweening vs. Soft skinning

Vertex tweening	Soft skinning
Un maillage par pose	Keyframing du squelette, un seul maillage pour toutes les animations
Déformations dues à l'interpolation linéaire	Pas de déformations
Dupliquer l'animation pour chaque modèle différent	Un seul squelette d'animation convient pour plusieurs skins différentes
Combinaison d'animations coûteuse (ensemble des vertices de la géométrie)	Combinaison d'animations simplifiée (combinaison des squelettes)
Sous-animations complexes à mettre en œuvre	Sous-animations très simples à mettre en œuvre

Animation réaliste

- Intégrer l'animation d'un acteur :
 - Le maillage est certes animé, mais comment le déplacer de manière réaliste dans l'espace ?
 - Comment faire pour établir des transitions correctes entre les différentes animations ?
 - Comment limiter le nombre total d'animations produites par les infographistes ?
 - Comment synchroniser des événements avec l'animation d'un acteur ?



Cinématique inverse

Il est impossible de générer toutes les animations pouvant gérer correctement tous les cas de figure :

→ **cinématique inverse** (IK – Inverse kinematics)

Méthode itérative :

- structure des squelettes d'animation : une chaîne de joints (articulations) et de bones,
- chaque articulation : des degrés de liberté (DOF – Degrees Of Freedom), fixent les limites en rotation (et éventuellement en translation),
- application de l'animation traditionnelle, puis on va chercher à placer le bout de la chaîne (ex: le pied au bout de la jambe) sur une position cible.

<https://www.youtube.com/watch?v=MvuO9ZHGr6k>

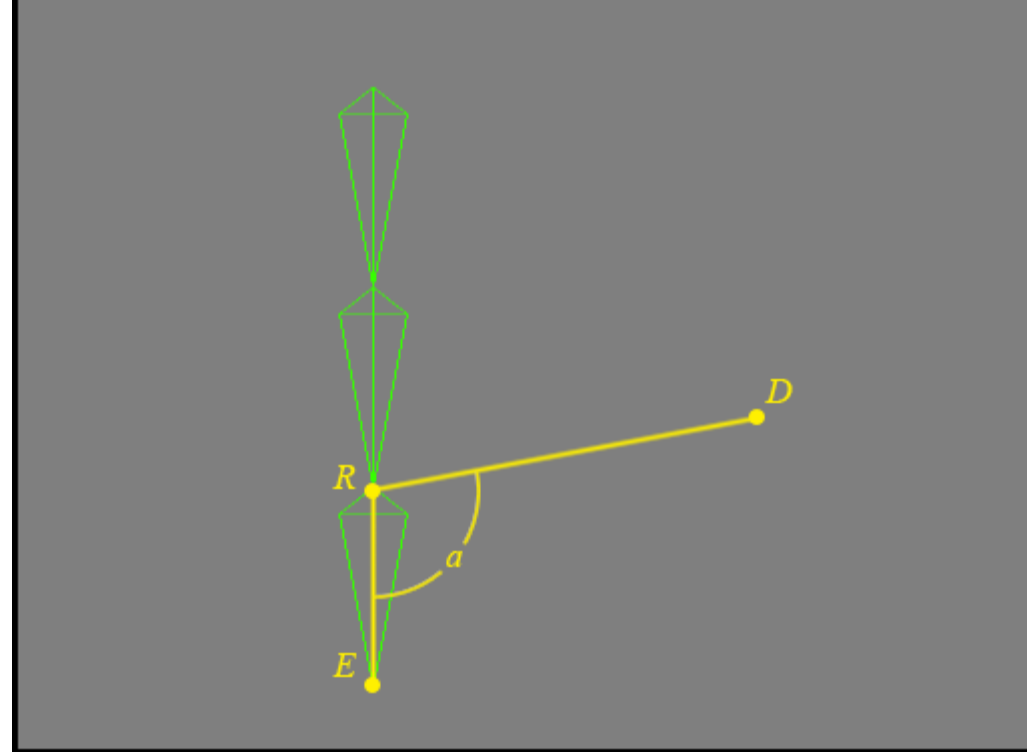
Pour chacune des articulations de la chaîne à partir de la source :

- **petite rotation** (en respectant les DOF) : approcher le bout de la chaîne articulaire de la position destination
- itérer jusqu'à ce que la position cible soit suffisamment proche ou que le système se soit stabilisé (pas de solution « exacte »).

Procédure de cinétique inverse est difficile : il peut y avoir plusieurs solutions possibles (ou aucune)

Résultats souvent très convaincants lorsque l'utilisation reste très localisée.

Fig5: Cyclic-Coordinate Descent (CCD)



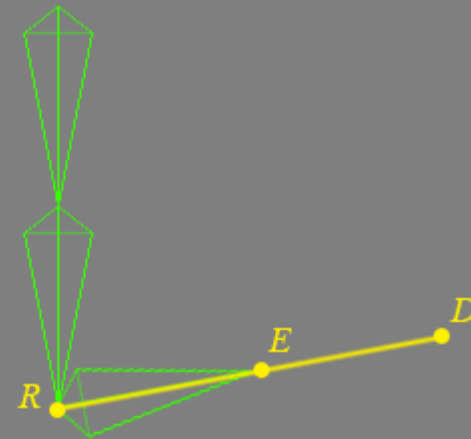
Pour chacune des articulations de la chaîne à partir de la source :

- **petite rotation** (en respectant les DOF) : approcher le bout de la chaîne articulaire de la position destination
- itérer jusqu'à ce que la position cible soit suffisamment proche ou que le système se soit stabilisé (pas de solution « exacte »).

Procédure de cinétique inverse est difficile : il peut y avoir plusieurs solutions possibles (ou aucune)

Résultats souvent très convaincants lorsque l'utilisation reste très localisée.

Fig5: Cyclic-Coordinate Descent (CCD)



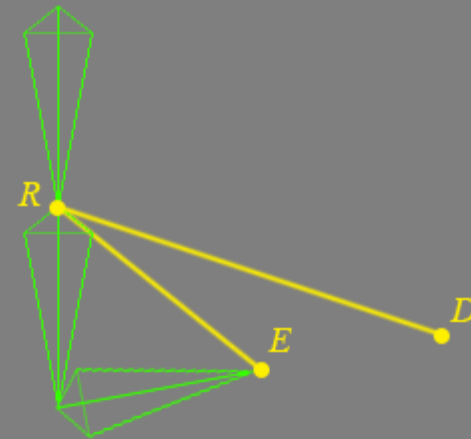
Pour chacune des articulations de la chaîne à partir de la source :

- **petite rotation** (en respectant les DOF) : approcher le bout de la chaîne articulaire de la position destination
- itérer jusqu'à ce que la position cible soit suffisamment proche ou que le système se soit stabilisé (pas de solution « exacte »).

Procédure de cinétique inverse est difficile : il peut y avoir plusieurs solutions possibles (ou aucune)

Résultats souvent très convaincants lorsque l'utilisation reste très localisée.

Fig5: Cyclic-Coordinate Descent (CCD)



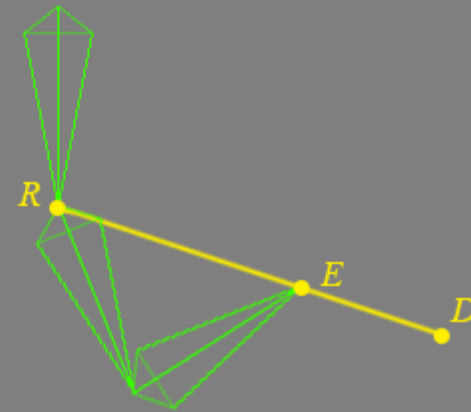
Pour chacune des articulations de la chaîne à partir de la source :

- **petite rotation** (en respectant les DOF) : approcher le bout de la chaîne articulaire de la position destination
- itérer jusqu'à ce que la position cible soit suffisamment proche ou que le système se soit stabilisé (pas de solution « exacte »).

Procédure de cinétique inverse est difficile : il peut y avoir plusieurs solutions possibles (ou aucune)

Résultats souvent très convaincants lorsque l'utilisation reste très localisée.

Fig5: Cyclic-Coordinate Descent (CCD)

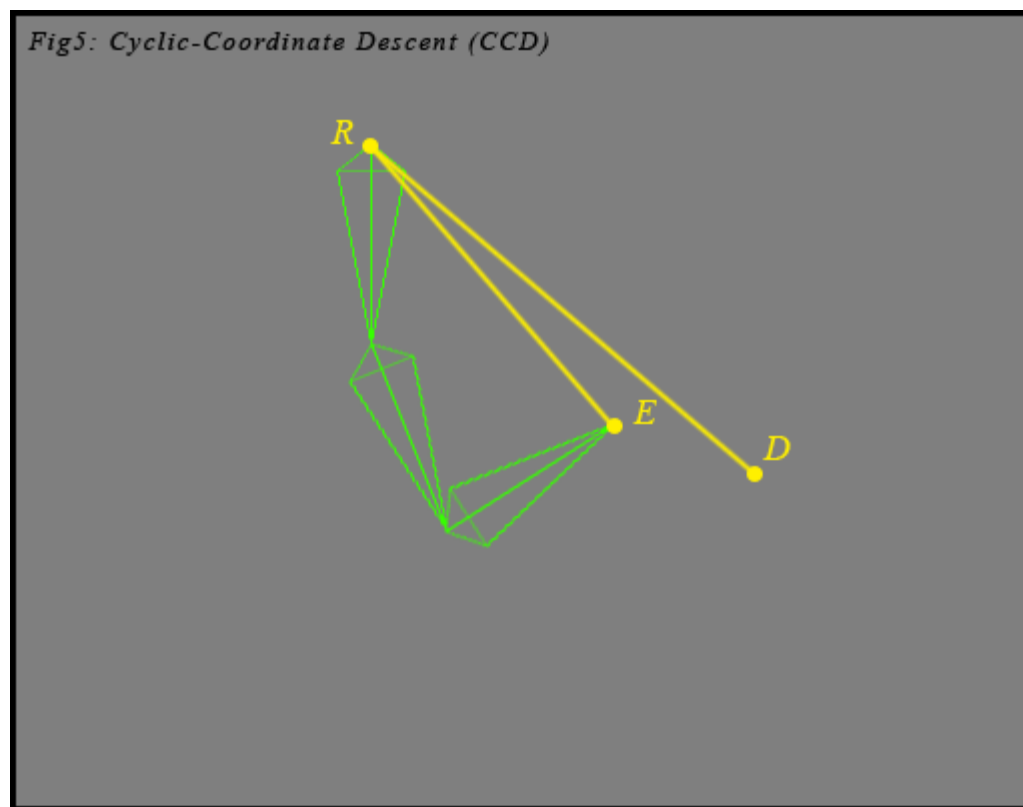


Pour chacune des articulations de la chaîne à partir de la source :

- **petite rotation** (en respectant les DOF) : approcher le bout de la chaîne articulaire de la position destination
- itérer jusqu'à ce que la position cible soit suffisamment proche ou que le système se soit stabilisé (pas de solution « exacte »).

Procédure de cinétique inverse est difficile : il peut y avoir plusieurs solutions possibles (ou aucune)

Résultats souvent très convaincants lorsque l'utilisation reste très localisée.

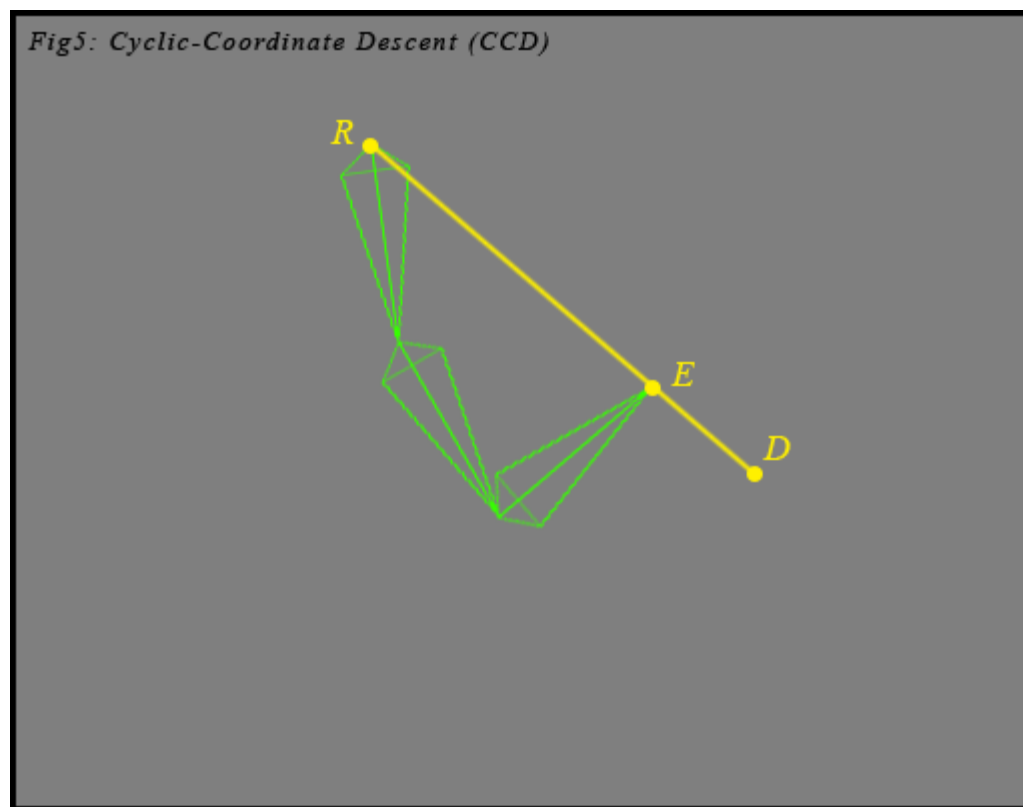


Pour chacune des articulations de la chaîne à partir de la source :

- **petite rotation** (en respectant les DOF) : approcher le bout de la chaîne articulaire de la position destination
- itérer jusqu'à ce que la position cible soit suffisamment proche ou que le système se soit stabilisé (pas de solution « exacte »).

Procédure de cinétique inverse est difficile : il peut y avoir plusieurs solutions possibles (ou aucune)

Résultats souvent très convaincants lorsque l'utilisation reste très localisée.



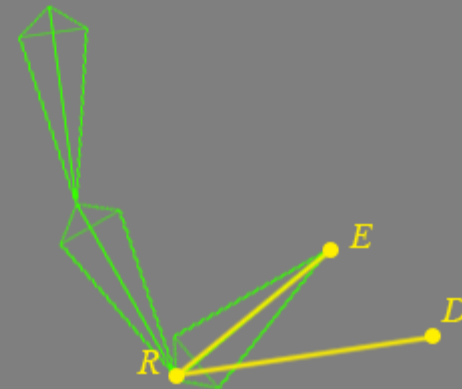
Pour chacune des articulations de la chaîne à partir de la source :

- **petite rotation** (en respectant les DOF) : approcher le bout de la chaîne articulaire de la position destination
- itérer jusqu'à ce que la position cible soit suffisamment proche ou que le système se soit stabilisé (pas de solution « exacte »).

Procédure de cinétique inverse est difficile : il peut y avoir plusieurs solutions possibles (ou aucune)

Résultats souvent très convaincants lorsque l'utilisation reste très localisée.

Fig5: Cyclic-Coordinate Descent (CCD)



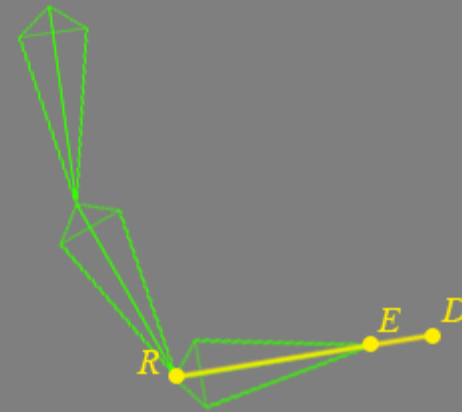
Pour chacune des articulations de la chaîne à partir de la source :

- **petite rotation** (en respectant les DOF) : approcher le bout de la chaîne articulaire de la position destination
- itérer jusqu'à ce que la position cible soit suffisamment proche ou que le système se soit stabilisé (pas de solution « exacte »).

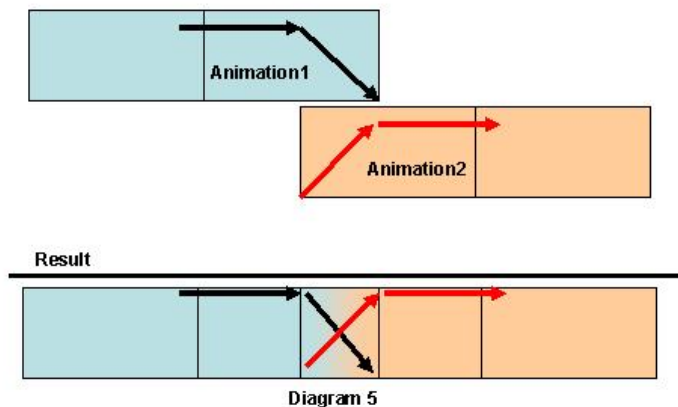
Procédure de cinétique inverse est difficile : il peut y avoir plusieurs solutions possibles (ou aucune)

Résultats souvent très convaincants lorsque l'utilisation reste très localisée.

Fig5: Cyclic-Coordinate Descent (CCD)



Blending d'animations



Blending d'animations

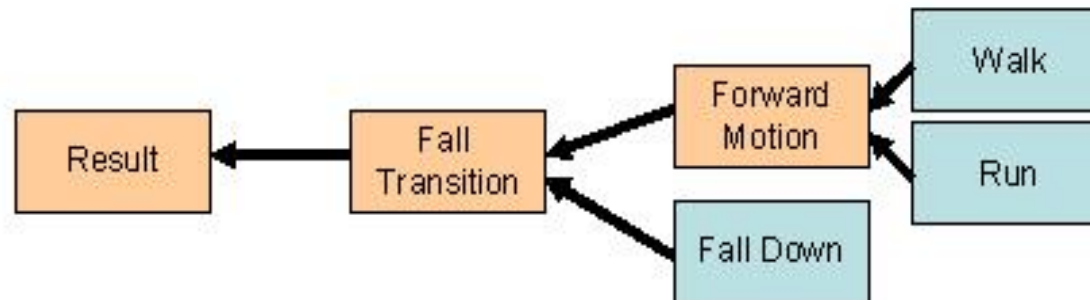
Comment gérer correctement les transitions entre deux animations (ex: marche/course/saut/chute), sachant que la transition peut intervenir à n'importe quel point dans l'animation ?

→ mixer plusieurs animations entre elles : le squelette d'animation va, le temps de la transition, **être une moyenne pondérée des deux animations** source et cible.

Blending d'animations

Gestionnaire d'animation capable de mixer plusieurs couches d'animation pour générer une animation complexe :

→ structure en arbre.



```
BlendLayer *forward_motion = new BlendLayer();
forward_motion->AddAnimation(WalkAnim, walk_weight);
forward_motion->AddAnimation(RunAnim, run_weight);

BlendLayer *fall_transition = new BlendLayer();
fall_transition->AddAnimation(FallAnim, fall_weight);
fall_transition->AddAnimation(forward_motion, motion_weight);

result_layer = new BlendLayer();
result_layer->AddAnimation(fall_transition, 1.0f);

result_layer->Blend();
```

combinaison des différentes couches d'animation

Blending d'animations

Gestionnaire d'animation capable de mixer plusieurs couches d'animation pour générer une animation complexe :

→ structure en arbre.

Animations peuvent être locales (ex: le joueur recharge son arme pendant sa course)

→ réduction du nombre total d'animation à produire

→ génération automatique de nouvelles animations : combinaisons de sous-animations au runtime.



Personnage de Drake (Uncharted) combine jusqu'à 25 animations.

Animation faciale

- L'animation faciale réaliste couvre plusieurs domaines de recherche.
- Dans sa version « temps réel », peut être implémentée dans les jeux, on notera les spécificités importantes suivantes:
 - Expressions réalistes
 - Synchronisation labiale (lipsync)



<https://www.youtube.com/watch?v=LjjmNoMx4eo>

Blendshapes

Animation faciale temps réel « réaliste » par « morph targets » ou « **blend shapes** » :

1. définition de **plusieurs maillages** représentant les expressions remarquables,
2. **moyenner la pondération** obtenue par vertex tweening de chacune de ces expressions relativement au modèle de base.

Cette méthode se basant sur une déformation du maillage réalisée manuellement en amont :

- expressions,
- déformations des muscles.



Exemples de blend shapes pour l'animation faciale

Blendshapes

Techniquement, la création des blend shapes est réalisée par les artistes à partir de la pose statique.

Déformation est très localisée, seuls les vertices déformés sont stockés
→ réduction de l'empreinte mémoire.

Côté animation, les morph targets sont appliquées sur le mesh **avant la phase de skinning**, de manière à pouvoir subir l'influence de l'animation globale.

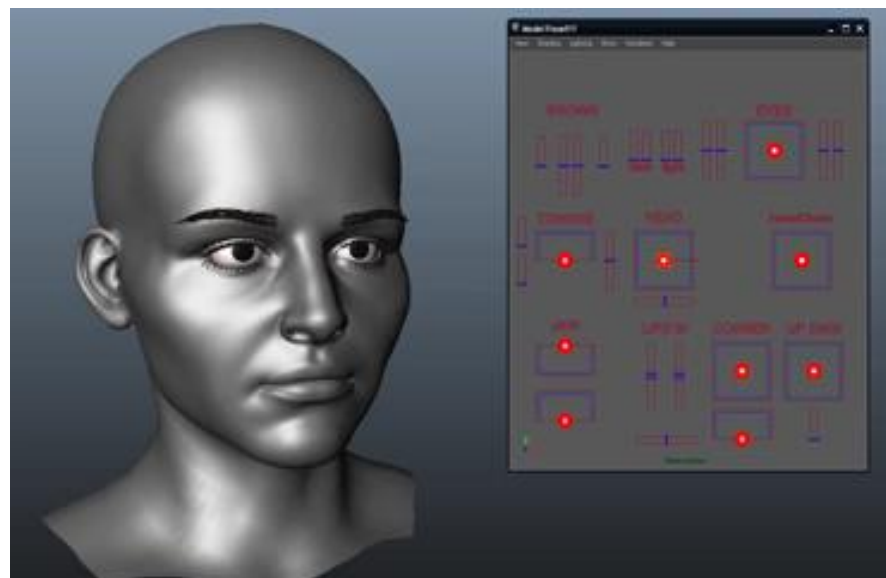


Blendshapes

Techniquement, la création des blend shapes est réalisée par les artistes à partir de la pose statique.

Déformation est très localisée, seuls les vertices déformés sont stockés
→ réduction de l’empreinte mémoire.

Côté animation, les morph targets sont appliquées sur le mesh **avant la phase de skinning**, de manière à pouvoir subir l'influence de l'animation globale.

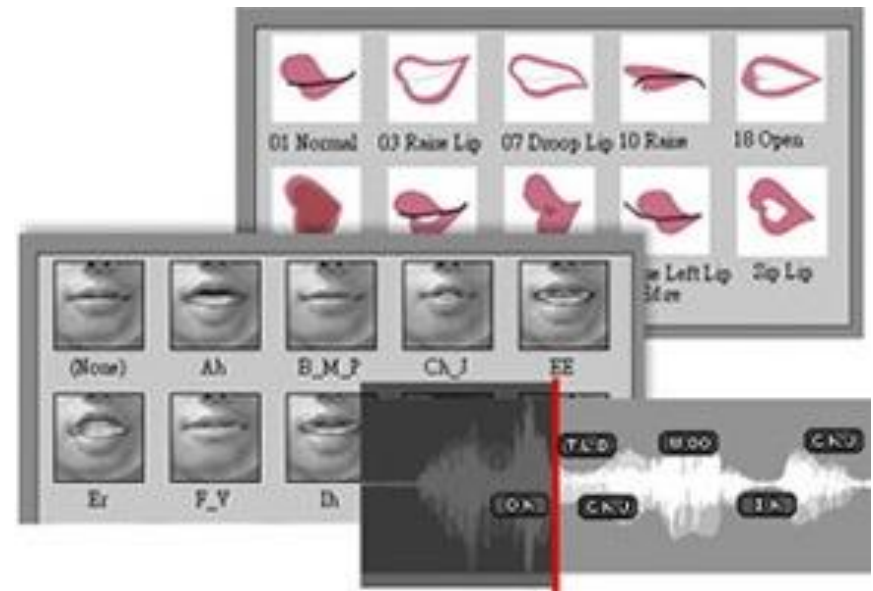


Synchronisation labiale (lipsync)

La mise en œuvre d'une synchronisation labiale se déroule en 2 phases:

L'infographiste modélise les blend shapes correspondant aux différentes formes de bouche (environ 9 par approximation)

- (1) A, I
- (2) E
- (3) F, V
- (4) C, D, G, J, K, N, S, T, Y, Z
- (5) L, T
- (6) O
- (7) U
- (8) W, Q
- (9) M, B, P (maillage de base)



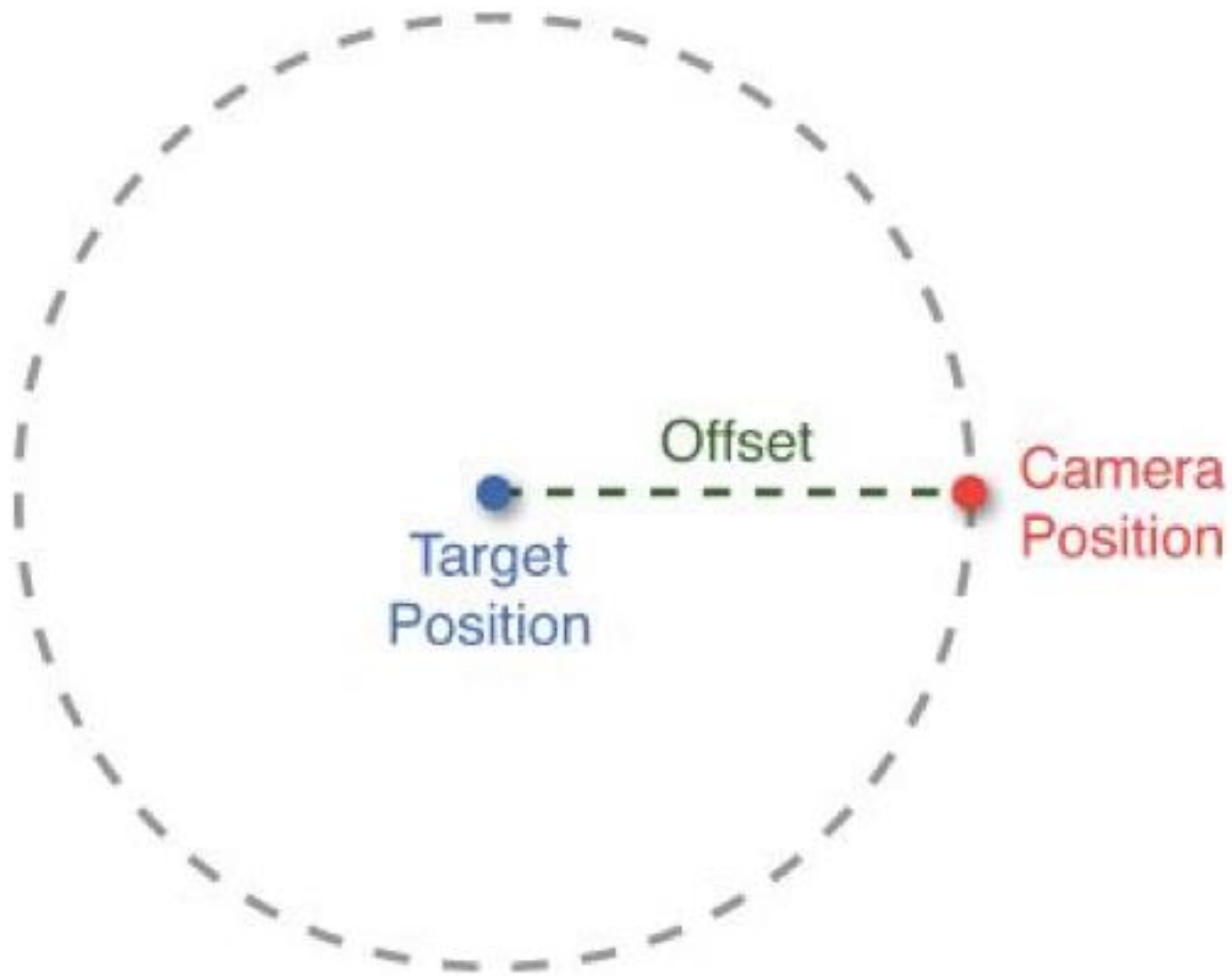
Analyse fréquentielle du texte à synchroniser, et extraction des phonèmes remarquables (à associer à nos blend shapes).

Générer une série de keyframes encodant l'animation à appliquer sur les blend shapes.

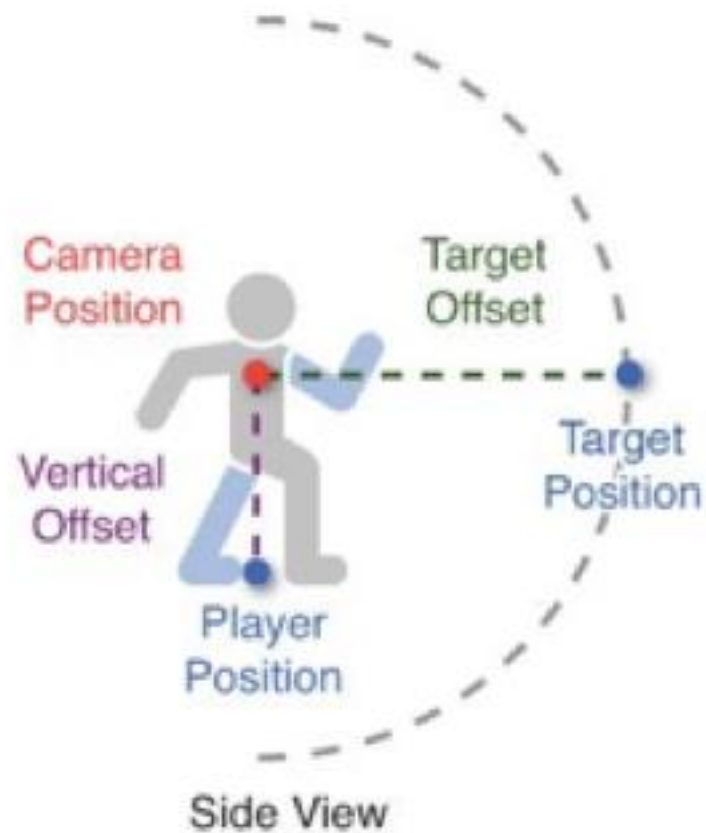
Animation de la caméra

- La gestion de la caméra est une partie importante dans une application 3D interactive
 - Elle doit suivre l'action et offrir le meilleur champ de vision possible
 - Elle doit correctement interagir avec l'environnement (collisions, masquages, effets spéciaux)
- On distingue deux types de caméra:
 - Caméra immersive (first person)
 - Caméra de suivi (tracking ou third person)

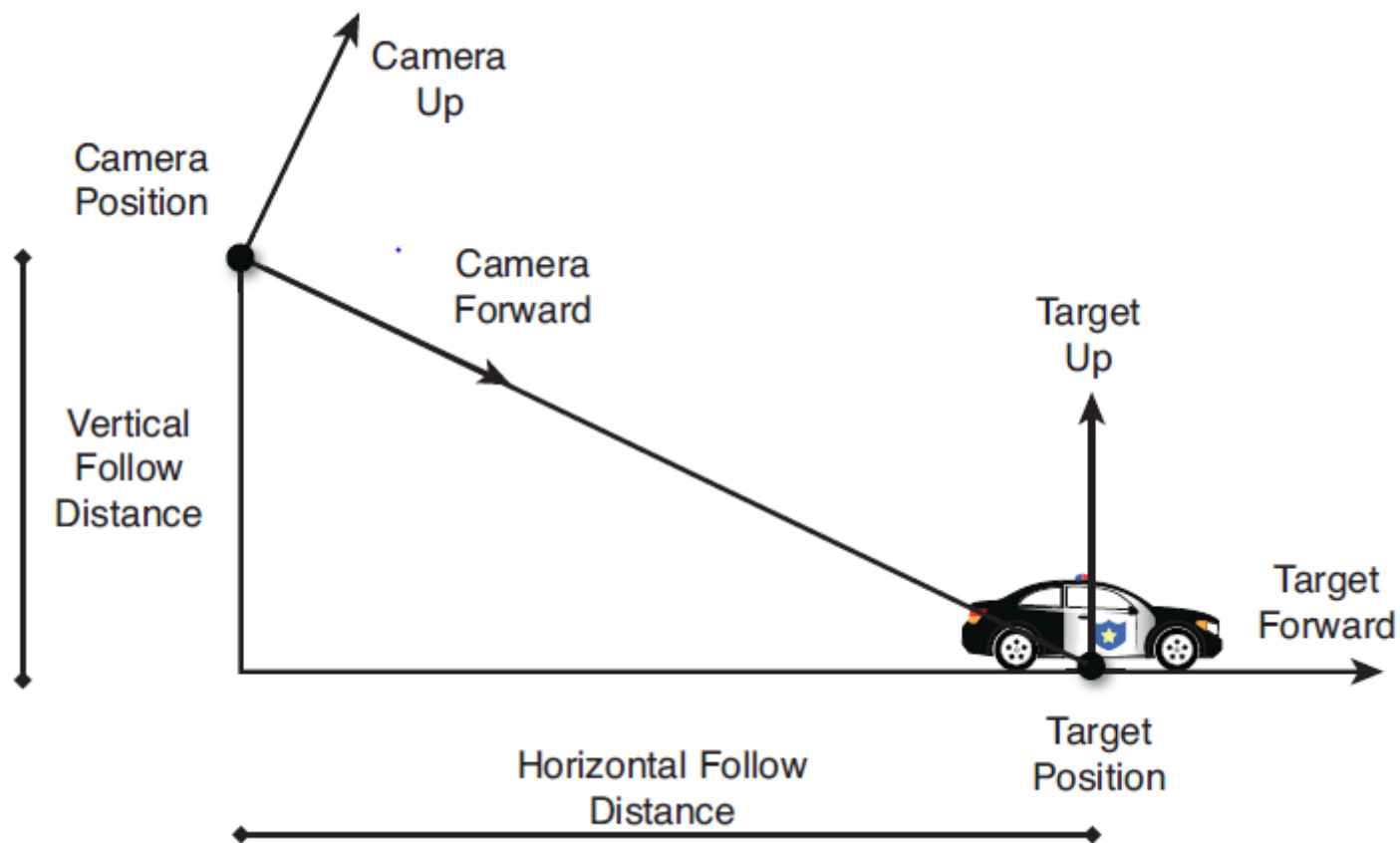
Orbit camera



First-person camera



Third-person camera



Animation de la caméra

- Objet caméra est souvent attaché à la hiérarchie (squelette) d'animation de l'acteur suivi (en position et/ou orientation)
→ subi directement les transformations induites par l'animation du personnage.
- Problème : données d'animation peuvent être bruitées (surtout motion capture), ou variations brutales de vitesse de objet suivi
→ mouvements directement retransmis à la caméra : comportement relativement rigide et non naturel.
- Réduction des artefacts : un modèle de caméra permettant d'amortir les imperfections
→ modèle masse-ressort pour appliquer les transformations à la caméra (la prochaine séance).

Gestion des collisions

Gestion des collisions avec les objets infranchissables de la scène (murs, parois, etc). La gestion des forces physiques appliquées à la caméra peut être intégrée directement dans la gestion des liaisons élastiques.

Gestion des objets pouvant masquer le champ visuel d'une caméra de type tracking, solutions :

- tester en permanence si un objet s'interpose entre la caméra et sa cible, et si c'est le cas se rapprocher de la cible pour se placer devant l'objet
- rendre transparents les objets interposés

- Optimisation du champ visuel

Pour suivre l'action au plus près, la caméra peut avoir à « anticiper » les déplacements de sa cible

Animation et cinématographie

Contrôle du cadrage et de la profondeur de champ

Variation dynamique de la profondeur de champ pour focaliser l'attention sur la cible

Variation dynamique de la focale pour renforcer les effets de vitesse (zooms compensés)

Ajout de bruit sur la position/orientation de la caméra pour simuler une caméra à l'épaule ou un effet d'aspiration

Ajout d'effets graphiques sur la caméra (pluie, buée, etc)

Pour aller plus loin

