Chapitre 4

ANIMATION BASÉE SUR L'INTERPOLATION

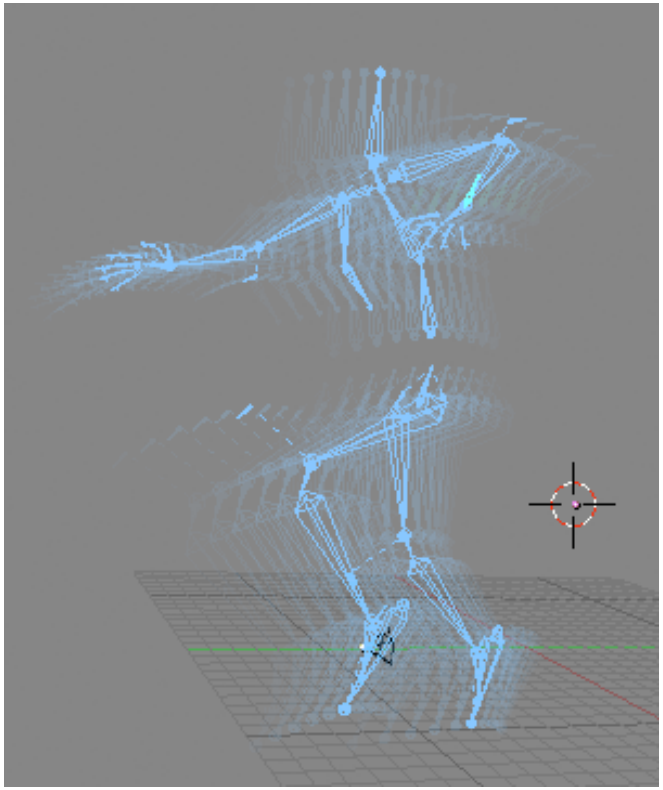
IMN504, Hiver 2022, Olivier Vaillancourt et Richard Egli
Université de Sherbrooke

PORTÉE DU CHAPITRE

- Les chapitres précédents étaient consacrés au mouvement en général.
- Ce chapitre introduit la notion de cadre clé (key-frame). Par la suite, la déformation de maillages (mesh) sera principalement abordée.
- La déformation de maillage se fera sur une animation structurée en temps réel appliquée à un modèle 3D.

PLAN DU CHAPITRE

- Types d'animation par interpolation
  - Animation par key-frame
  - Animation hiérarchique par squelette
    - Joints
    - Ossature (Bones)
    - Variables et contraintes d'articulations
- Méthodes de skinning
- Langages d'animation
  - Orientés artistes
  - Langages de programmation
  - Variables d'articulation
  - Langages graphiques
  - Langages basés acteurs



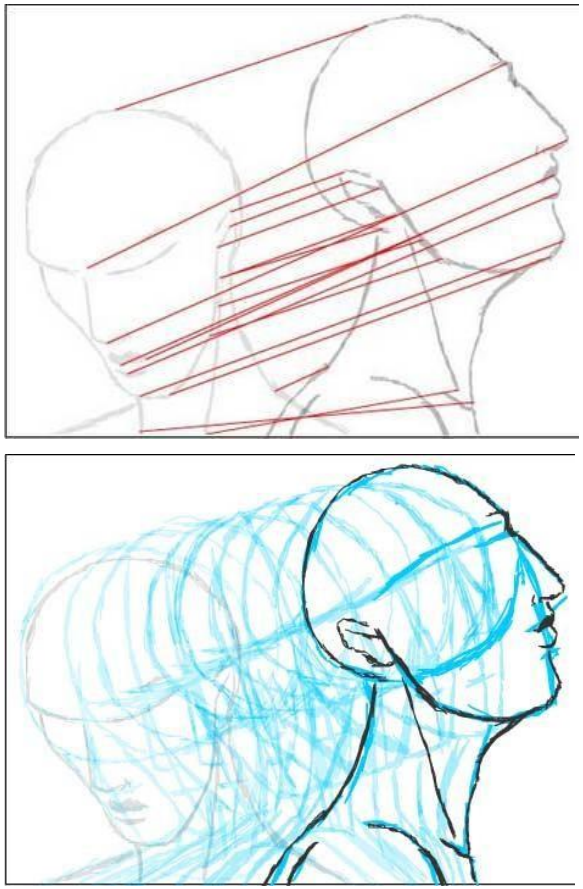
BlenderArtists.org, 2008

NOTION DE CADRE CLÉ (KEYFRAME)

- En animation 2D classique, un cadre clé est une **image** de l'animation déterminant une pose d'un objet/personnage.

- Les images entre les cadres clés sont appellées les "entre deux" ou "in-between".

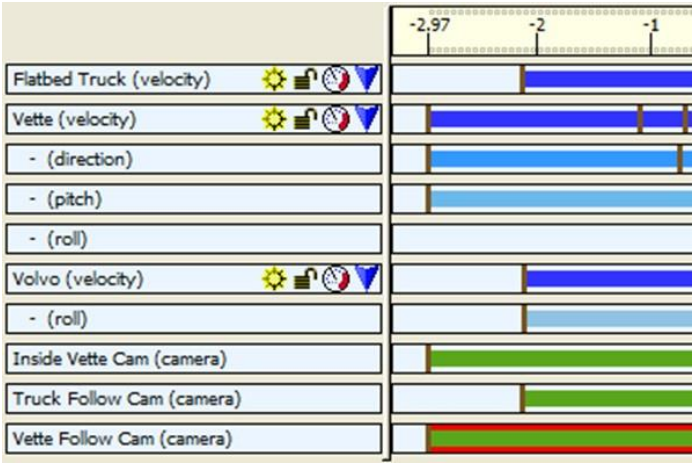
  - Les in-between sont habituellement générées par des interpolations



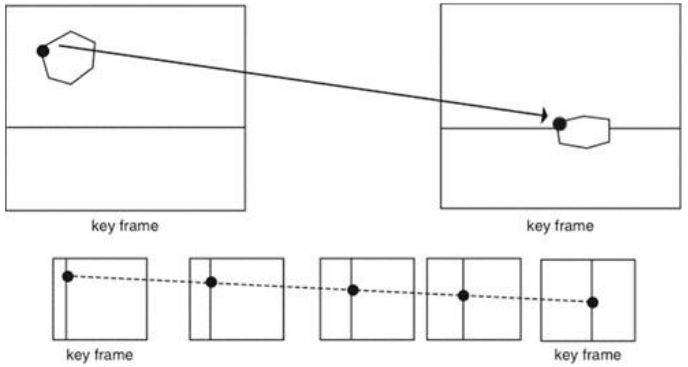
Adrien-Luc Sanders 2005

NOTION DE CADRE CLÉ (KEY-FRAME)

- En animation par ordinateur, un cadre clé définit tout moment où une variable d'une animation n'est pas interpolée. Il est spécifié par l'animateur.
  - Les différentes variables peuvent avoir différents cadres clés.
    - Position
    - Couleur
    - Rotation autour d'un axe
    - Etc.
  - Un cadre clé ne correspond plus obligatoirement à une pose globale d'une animation mais plutôt à un état fixé d'une variable.



MapScenes System, 2008



Parent, R. : "Computer Animation : Algorithms and techniques", third edition, Morgan Kaufmann, Fig 4.2

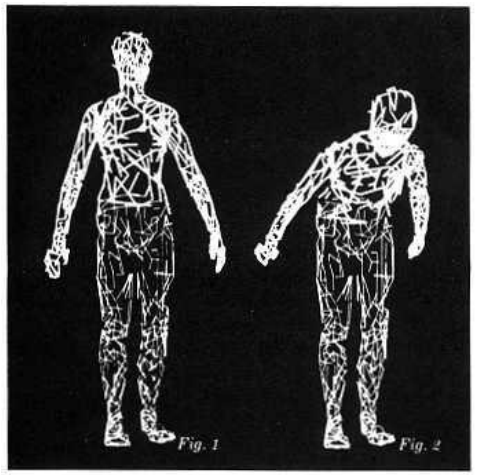
ANIMATION PAR KEYFRAME CLASSIQUE SUR ORDINATEUR

ANIMATION PAR KEY-FRAME CLASSIQUE SUR ORDINATEUR

- Chaque image de notre animation est stockée dans une unité complète, séparée du reste.

SNK, Metal Slug, 1999

  - En 2D on parle surtout de “Sprite”
    - Images 2D affichées en séquence.
  - En 3D on parle d’une suite de maillages (mesh)
    - Suite de modèles 3D rendus en séquence.
    - Peut être aussi utilisé en 2D.

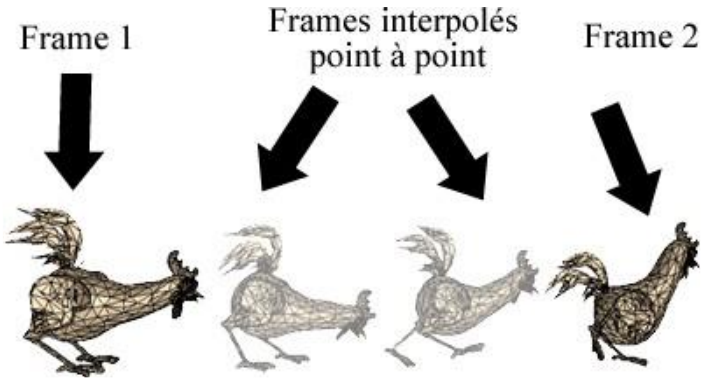


From keyframe to keyframe: computerized St. Catherine takes a bow.

A. Rebecca, Nisselson J. : “Computers who dance”, Atari Archive, 1983

ANIMATION PAR KEY-FRAME CLASSIQUE SUR ORDINATEUR

- Variation pour l'animation par key-frame classique en 3D:
  - Le modèle contient un nombre de frame réduit pour l'animation.
  - Chaque vertex est interpolé un à un entre les images pour combler les trous et produire l'animation.



ANIMATION PAR KEY-FRAME CLASSIQUE SUR ORDINATEUR

- Avantages
  - Simple
    - Simplement charger chaque maillage et les afficher séquentiellement.
  - Rapide
    - Une fois le maillage chargé, simplement l'afficher, ne demande aucun calcul supplémentaire.
  - Haute fidélité
    - Le modèle rendu a exactement l'air du modèle créé dans le logiciel d'animation.

ANIMATION PAR KEY-FRAME CLASSIQUE SUR ORDINATEUR

- Désavantages:
  - Mémoire
    - Un maillage séparé pour chaque cadre d'animation prend beaucoup d'espace.
  - Peu/Pas d'interaction avec l'animation
    - Aucune structure simplifiée pour faciliter l'interaction du modèle animé avec son environnement. (Pas vraiment possible d'ajouter des physiques ou de la cinématique inverse efficacement.)
    - Tout doit être fait d'avance (transitions entre les animations, actions, etc.)

ANIMATION PAR KEY-FRAME CLASSIQUE SUR ORDINATEUR

- Applications:
  - Applications 3D en temps réel
    - C.à.d.: Jeux vidéos principalement
  - Très populaire à la fin des années 1990
    - Émergence des cartes vidéo 3D
    - Ordinateurs manquaient de puissance pour d'autres types d'animation
  - Reprise de la popularité avec les consoles de jeu vidéo portables, téléphones cellulaires, etc.

De haut en bas: Quake 2, 1997; Animal Crossing, Wild World, 2005; Super Monkey ball (iPod)2008

ANIMATION PAR KEY-FRAME CLASSIQUE SUR ORDINATEUR

- En pratique:
  - Le modèle est habituellement créé/animé avec un squelette dans un logiciel de modélisation 3D.
  - Il est converti en une série de key-frames 3D représentant les images de l'animation lors de l'exportation.

ANIMATION PAR KEY-FRAMES INTERPOLÉS

ANIMATION PAR KEY-FRAMES INTERPOLÉS

- Si on exclut l'animation classique, en animation par ordinateur un key-frame se définit comme étant:
  - Le frame où l'état d'une variable d'animation est fixé.

- Exemples de variables:
  - Position
  - Orientation
  - Couleur
  - Échelle
  - Propriété d'un matériaux
  - [...]

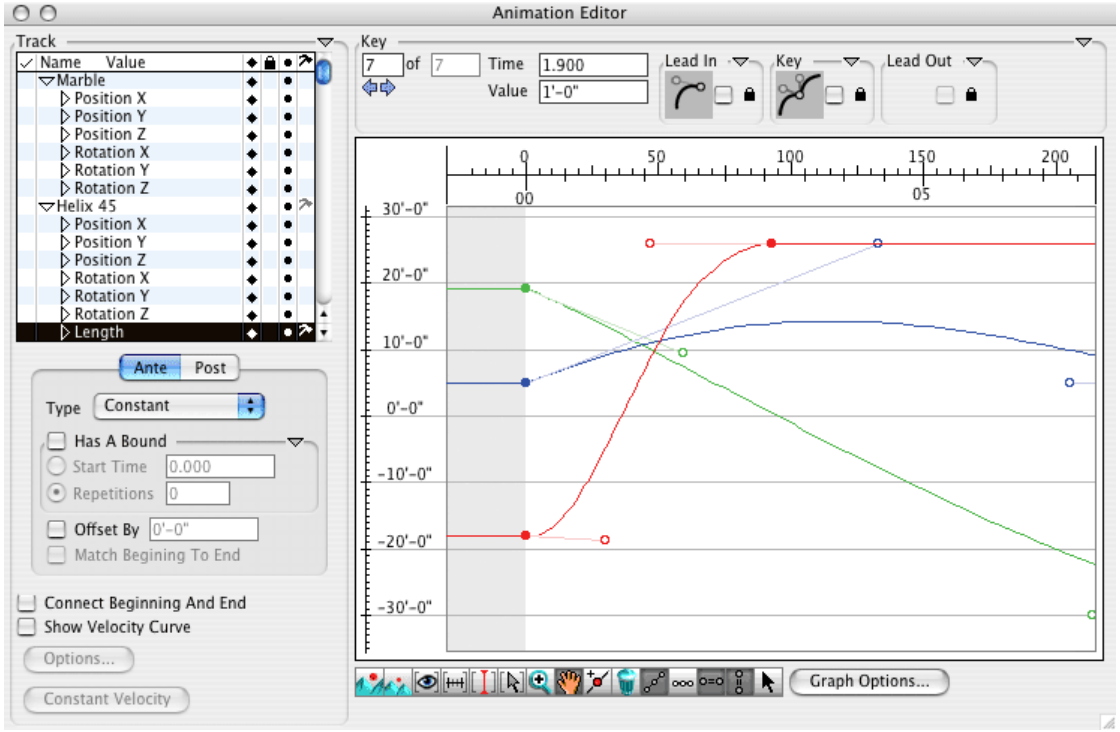
- On fixe le temps soit en termes direct (secondes, minutes, etc.) ou en termes de nombre d'images.

ANIMATION PAR KEY-FRAMES INTERPOLÉS

- Chaque variable d'animation possède sa propre ligne du temps.
  - On nomme cette ligne "piste" ou "track" d'animation.
  - Les différentes valeurs contenues sur les key-frames d'une piste d'animation sont interpolées pour générer l'animation en soit.

ANIMATION PAR KEY-FRAMES INTERPOLÉS

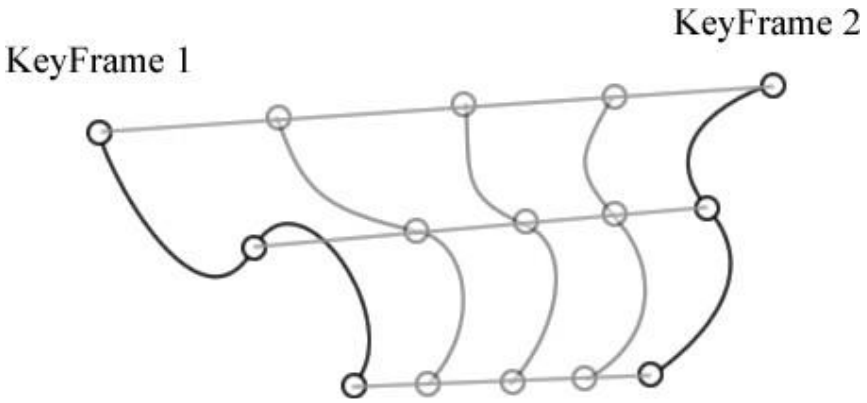
- Lorsqu'on combine les différentes valeurs contenues sur les différentes pistes, on obtient notre animation globale.



Form-Z 6.0, AutoDesSys, Inc, 2008

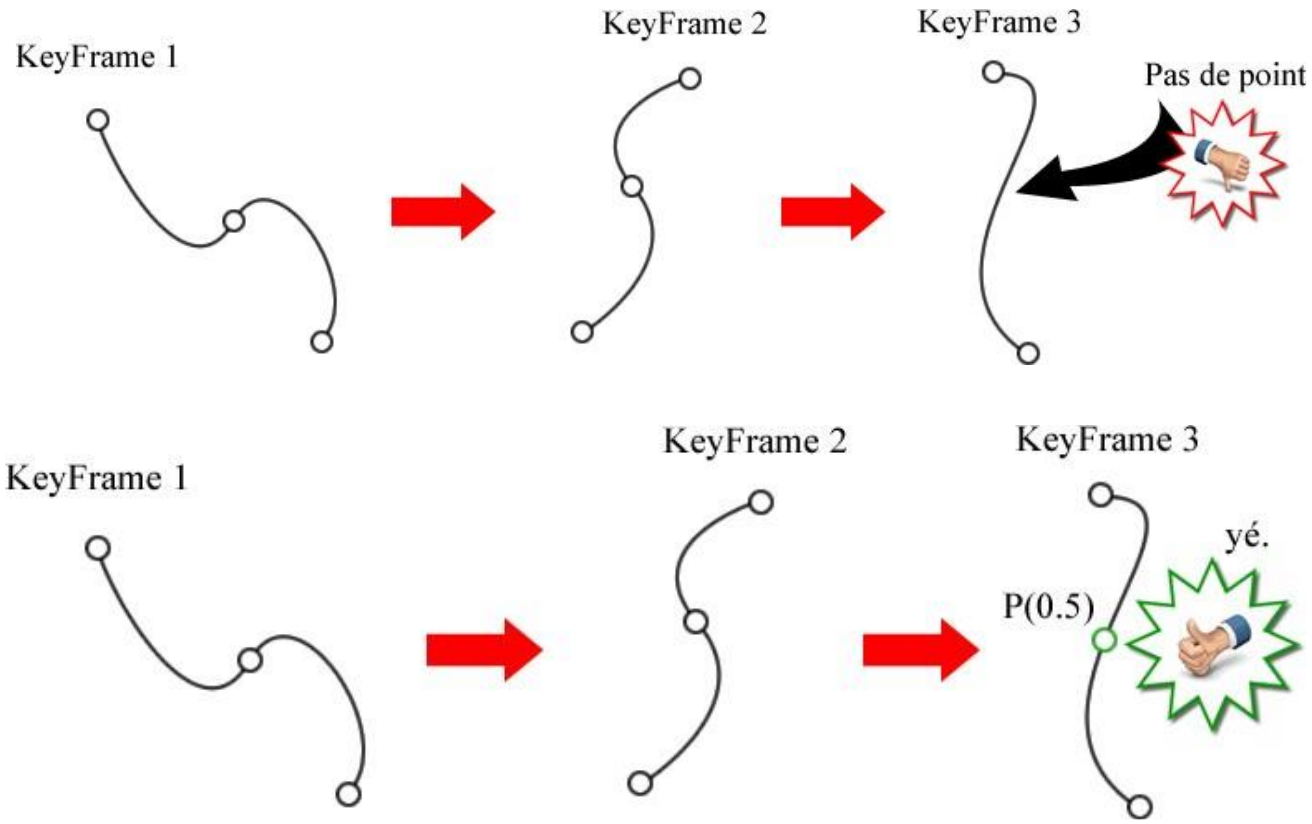
ANIMATION PAR KEY-FRAMES INTERPOLÉS

- Cas particulier d'interpolation par key-frame: Interpolation de segments
  - Si le nombre de points de contrôle est égal entre chaque key-frame:
    - On peut interpoler linéairement les points de contrôle et recréer le courbe à chaque image.



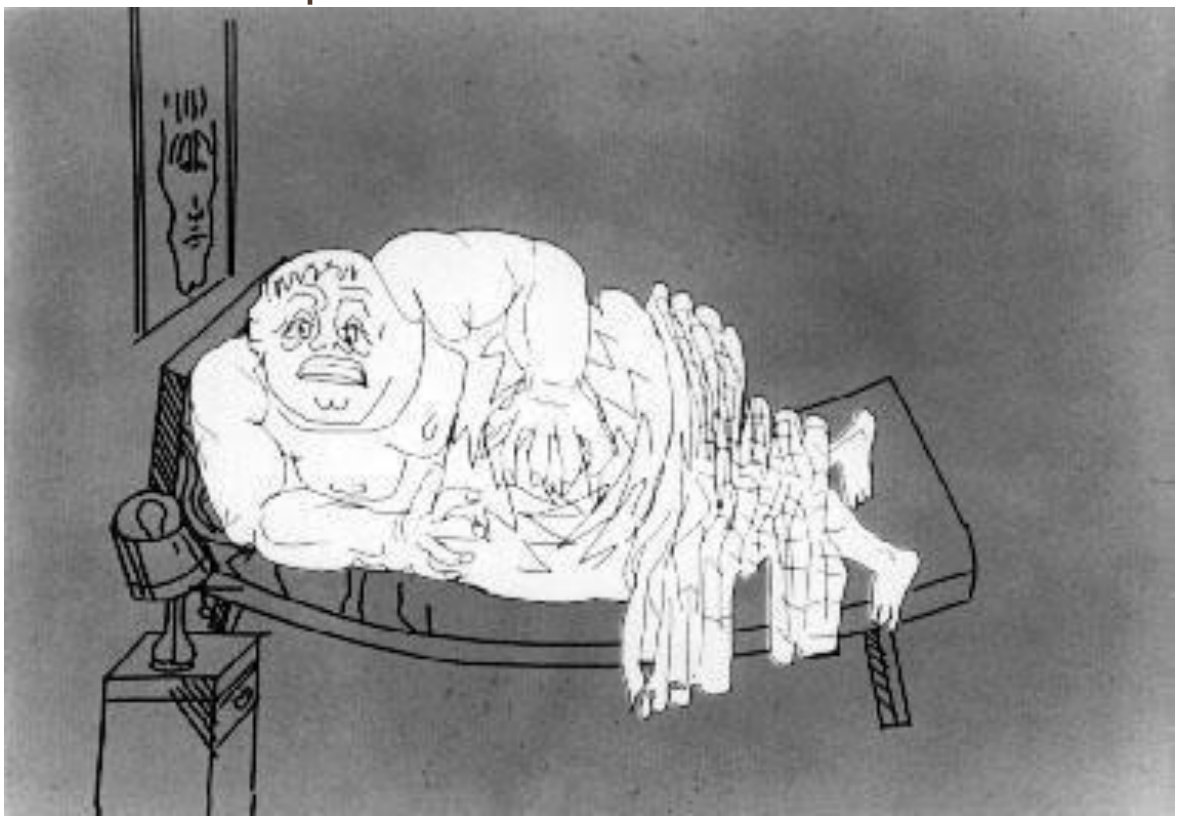
ANIMATION PAR KEY-FRAMES INTERPOLÉS

- Si d'un key-frame à l'autre le nombre de points de contrôle change
  - On peut générer des points intermédiaires manuellement en évaluant la fonction de notre courbe puis procéder avec l'interpolation.



ANIMATION PAR KEY-FRAMES INTERPOLÉS

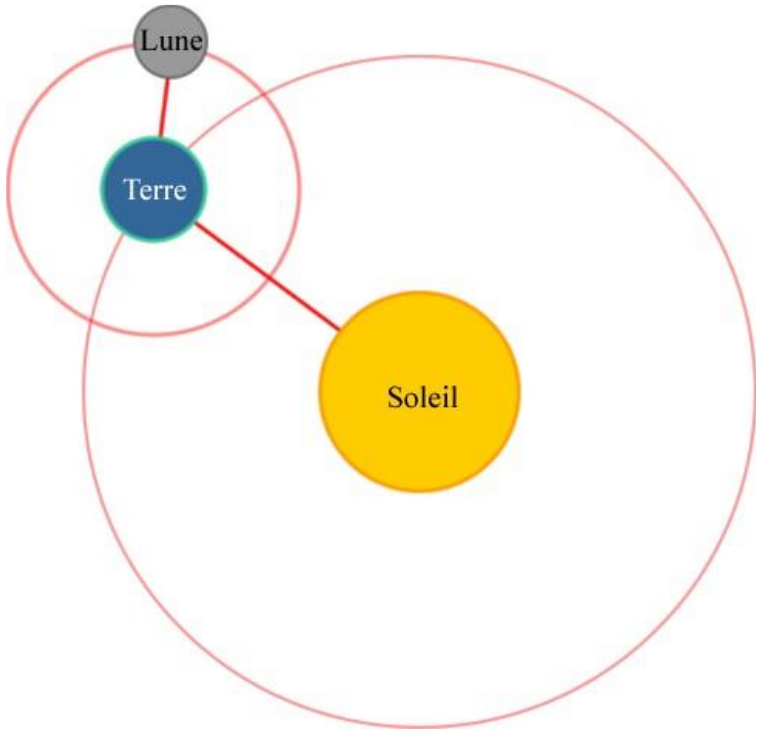
- Exemple: "*Hunger*", par Peter Foldes et René Jodoin,Office National du Film du Canada, 1973.
  - Première animation par ordinateur nominée aux oscars.



ANIMATION HIÉRARCHIQUE PAR SQUELETTE

MODÉLISATION HIÉRARCHIQUE DE L'ANIMATION (5.1)

- Consiste à exprimer l'animation d'un objet en fonction de transformations relatives à celles d'un autre objet.
- Par exemple: Modélisation hiérarchique d'un système solaire. (simplifié)



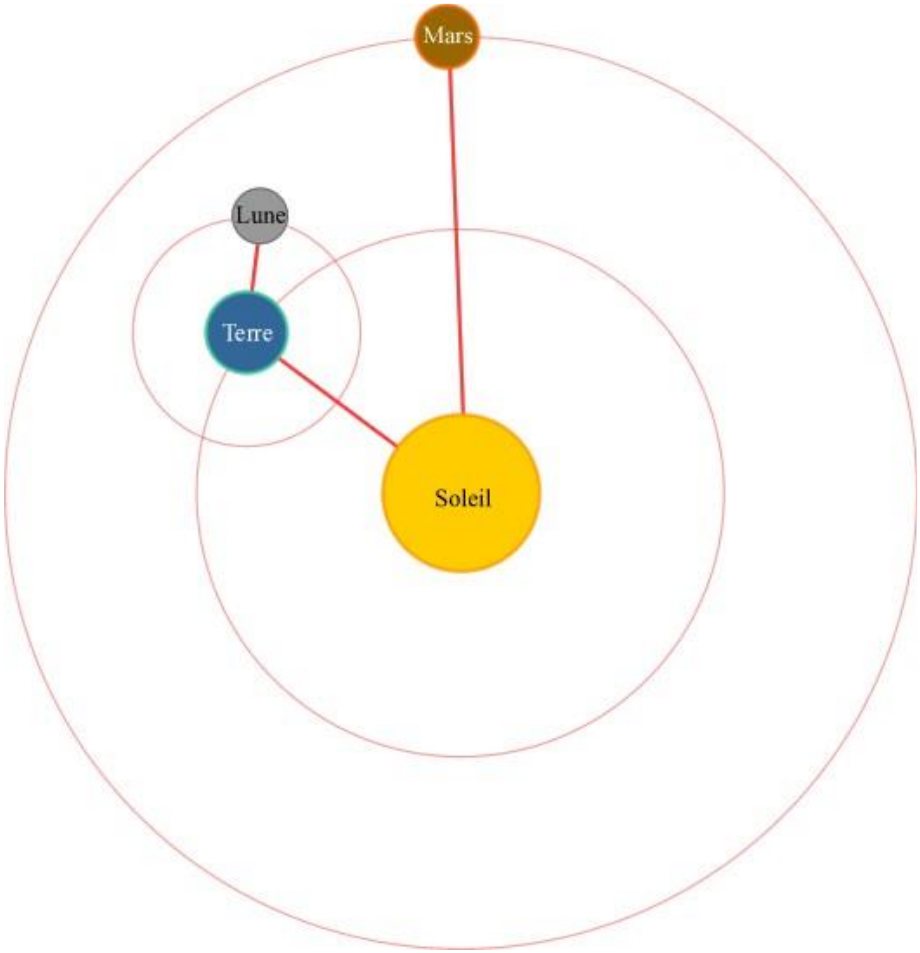
MODÉLISATION HIÉRARCHIQUE DE L'ANIMATION

- Vocabulaire:
  - Chaque objet dans la scène est appellé un **noeud**.
  - Si la transformation d'un noeud *a* dépend d'un noeud *b*, on dit que *a* est **l'enfant** de *b*.
  - Le noeud dont un enfant dépend est appellé son **parent**.
  - Un noeud n'ayant aucun parent est appellé la **racine**.
  - Un noeud n'ayant aucun enfant est une **feuille**.
  - Un noeud peut avoir **plusieurs enfants** mais **un seul parent**.

MODÉLISATION HIÉRARCHIQUE DE L'ANIMATION

- Exemple:

- Soleil → Racine
- Lune et Mars → Feuilles
- Terre → Parent de Lune
- Lune → Enfant de Terre
- Soleil → Parent de Terre et mars
- Etc.

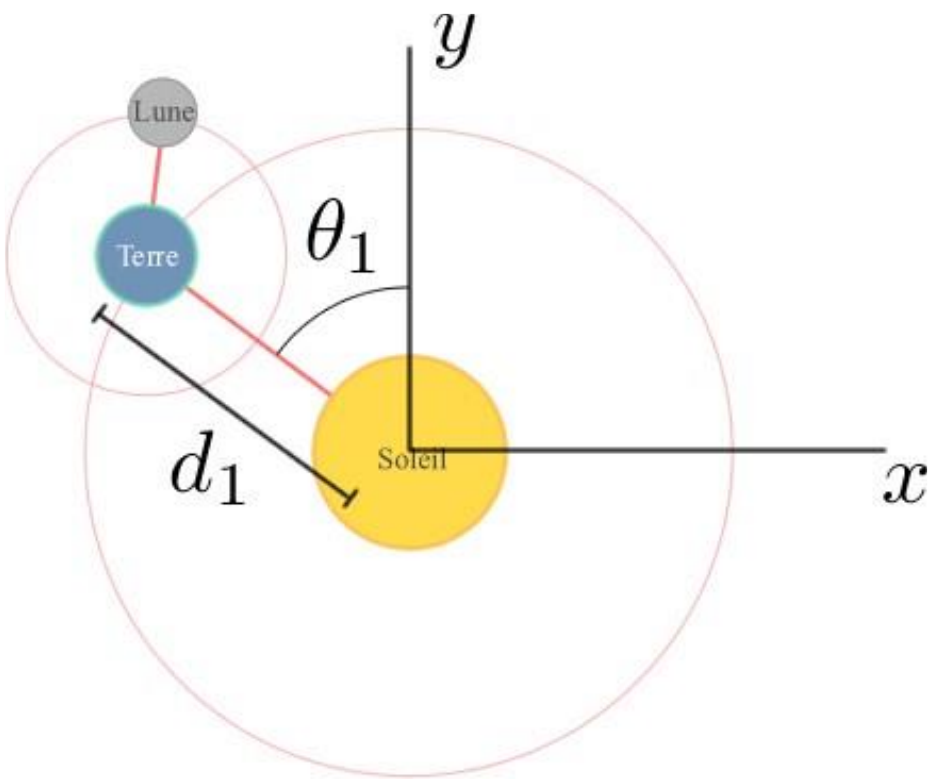


MODÉLISATION HIÉRARCHIQUE DE L'ANIMATION

- En exprimant les transformations d'un noeud par rapport à son parent, on contraint son positionnement ou son orientation par rapport à celui-ci.
  - Contraintes de distance (translation)
  - Contraintes d'orientation (rotation)

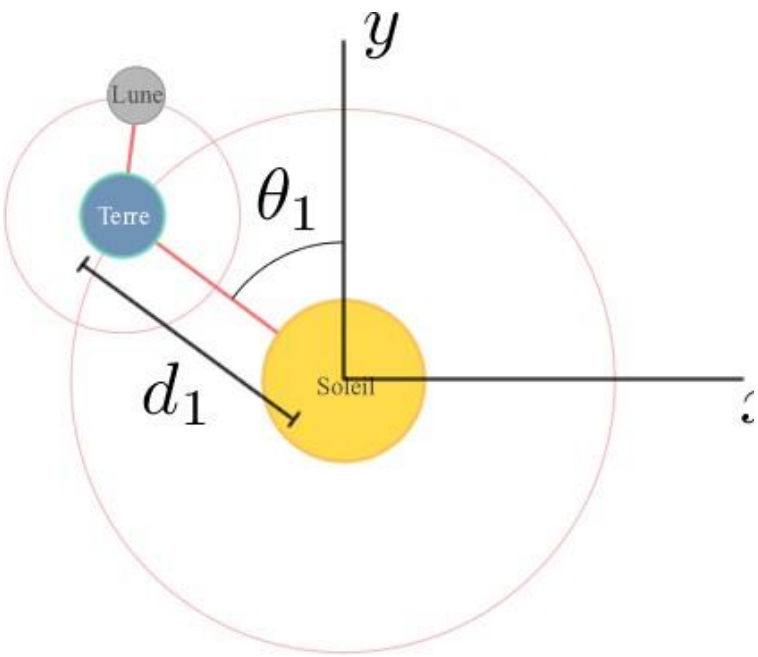
MODÉLISATION HIÉRARCHIQUE DE L'ANIMATION

- La dépendance entre les transformations permet de les simplifier de façon individuelle.
  - On caractérise cette simplification comme étant une **réduction de dimensionalité**.
- **Exemple:**
  La position d'une planète/lune se définie par une distance et un angle par rapport au corps autour duquel elle tourne. (On sauve la complexité requise pour calculer la position de l'objet de façon absolue).

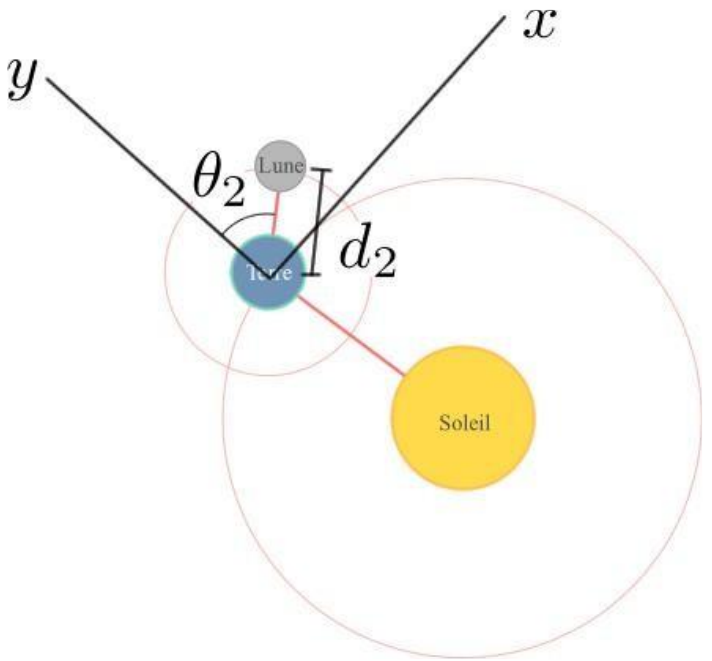


MODÉLISATION HIÉRARCHIQUE DE L'ANIMATION

- Chaque noeud possède son propre système de coordonnées locales.



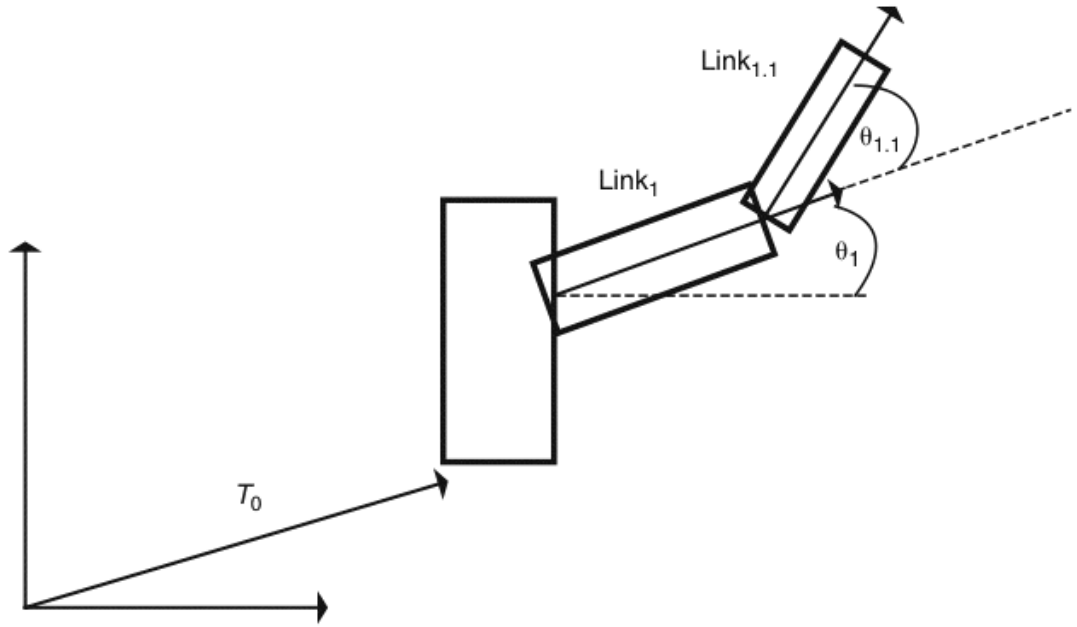
Système local du Soleil, la position de la Terre étant exprimée par rapport celui-ci.



Système local de la terre, la position de la lune étant exprimée par rapport à celui-ci

MODÉLISATION HIÉRARCHIQUE DE L'ANIMATION

- Pour déterminer la transformation finale d'un noeud:
  - On concatène les transformations en partant de la racine jusqu'au noeud en question.



Rick Parent, fig 5.11

$$Transf_{1.1} = T_0 T_1 R_1(\theta_1) T_{1.1} R_{1.1}(\theta_{1.1})$$

MODÉLISATION HIÉRARCHIQUE DE L'ANIMATION

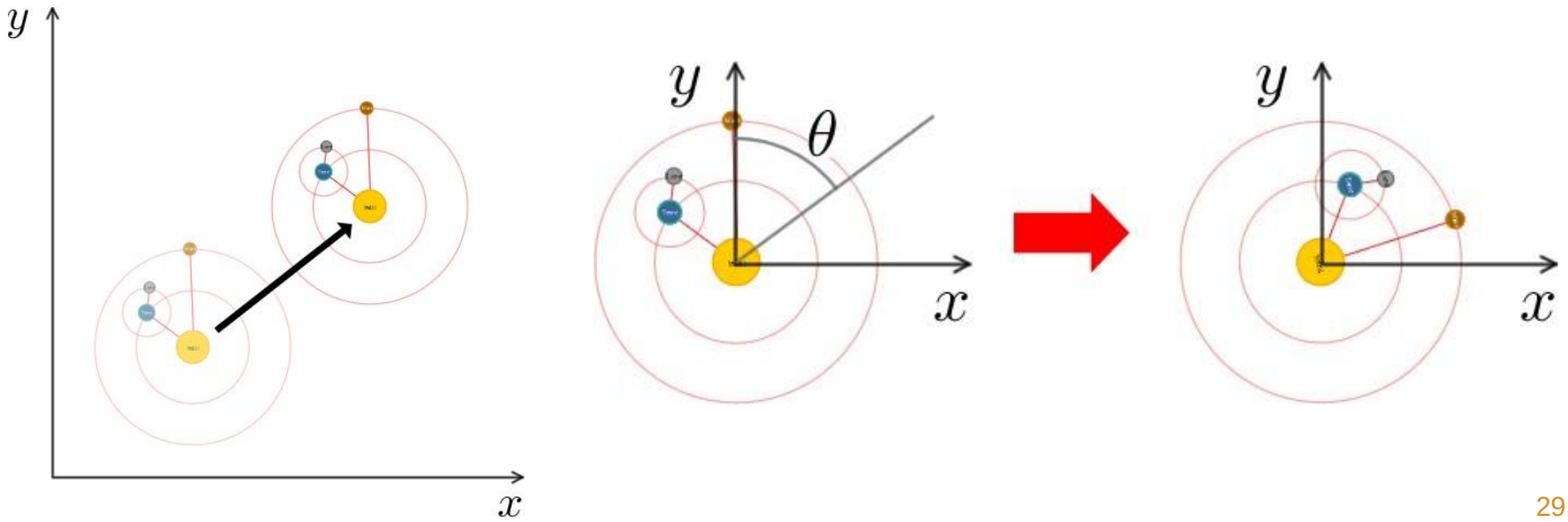
- Une fois qu'on a obtenu notre transformation globale, transformer les points associés au noeud en question se fait simplement.
  - Ex : (transformation du point V_1 associé au noeud 1.1)

$$Transf_{1.1} = T_0 T_1 R_1(\theta_1) T_{1.1} R_{1.1}(\theta_{1.1})$$

$$V_1' = (Transf_{1.1})(V_1)$$

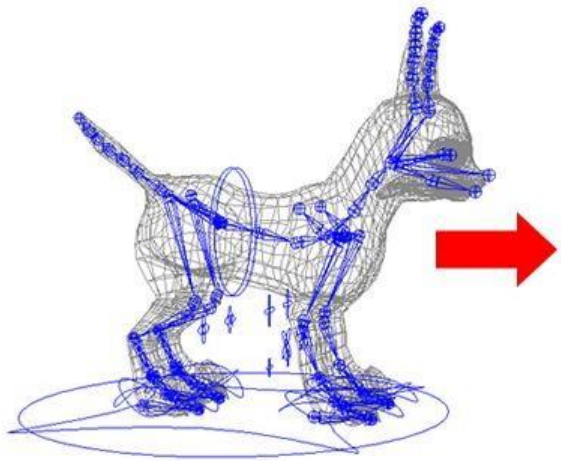
MODÉLISATION HIÉRARCHIQUE DE L'ANIMATION

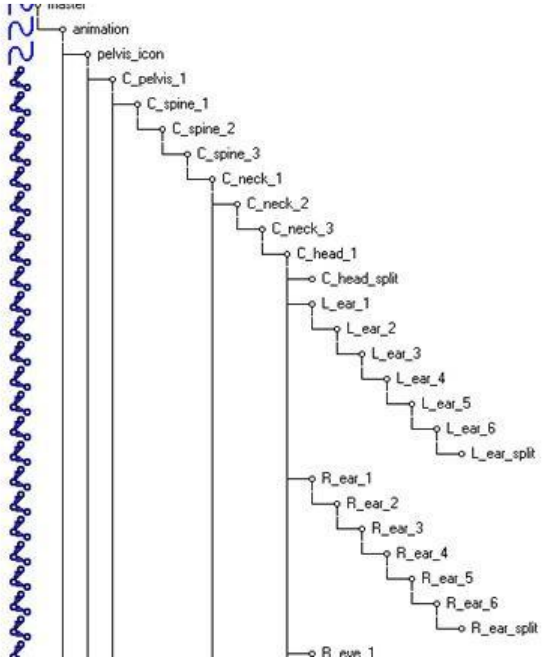
- Puisque les transformations partent du noeud racine, celui-ci est exprimé par défaut en espace monde.
  - Toute transformation appliquée sur la racine se répercutent sur toute la hiérarchie.

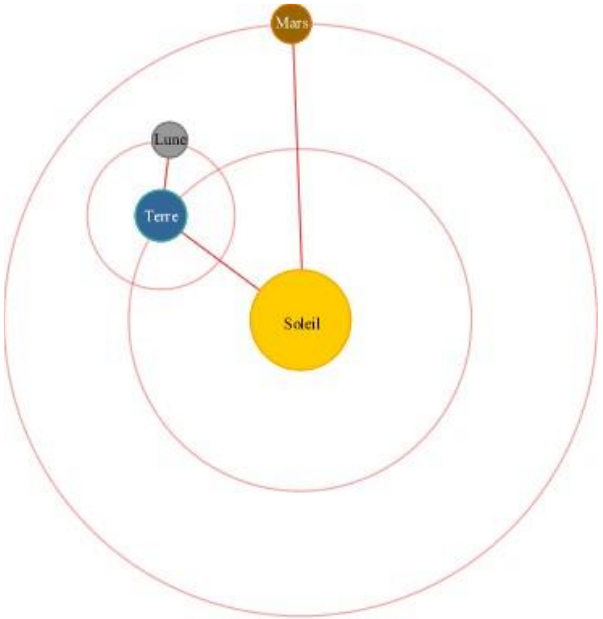


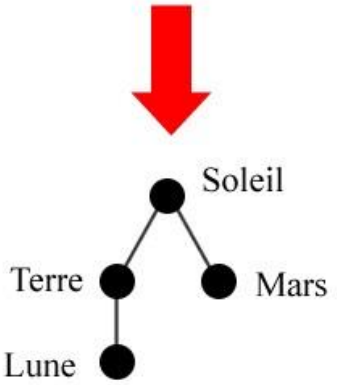
MODÉLISATION HIÉRARCHIQUE DE L'ANIMATION

- La hiérarchie entre nœuds peut être représentée sous la forme d'un graphe orienté acyclique.







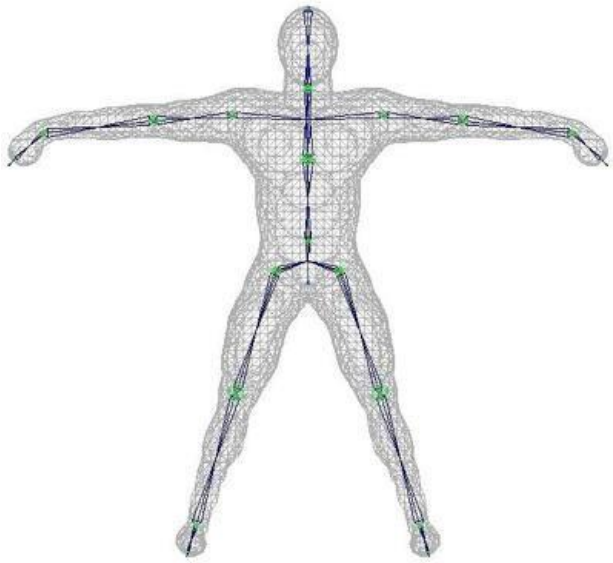


SQUELETTES D'ANIMATION

SQUELETTES D'ANIMATION

- Comme on vient de le voir:
  - La modélisation hiérarchique permet entre autre de modéliser des systèmes d'objets distincts.
  - Une autre approche est de modéliser une hiérarchie à l'intérieur même d'un objet pour gérer de façon localisée sa déformation.

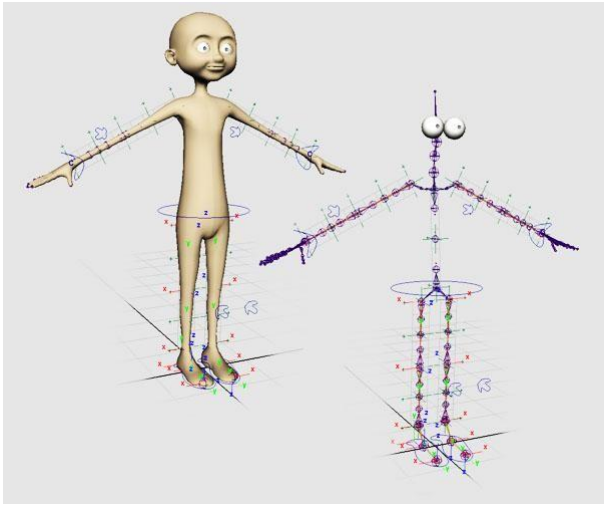
Linksquare, 2008



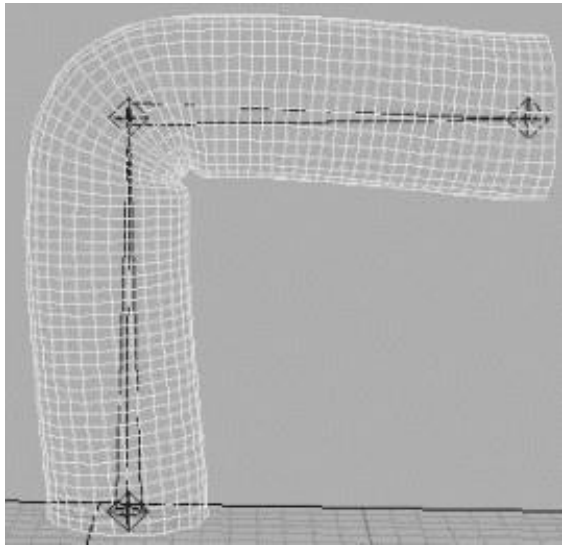
Frank A. Rivera, 1998

SQUELETTES D'ANIMATION

- Le modèle 3D contenant la hiérarchie peut être perçu comme une enveloppe.
  - Les différents points (vertices) qui composent le modèle sont reliés à différents noeuds de la hiérarchie.
  - L'association d'un point à un élément de la hiérarchie est appelé "skinning", nous verrons des méthodes de skinning plus en détail vers la fin du chapitre.



"Lucien", Mathias Aubry, 2008



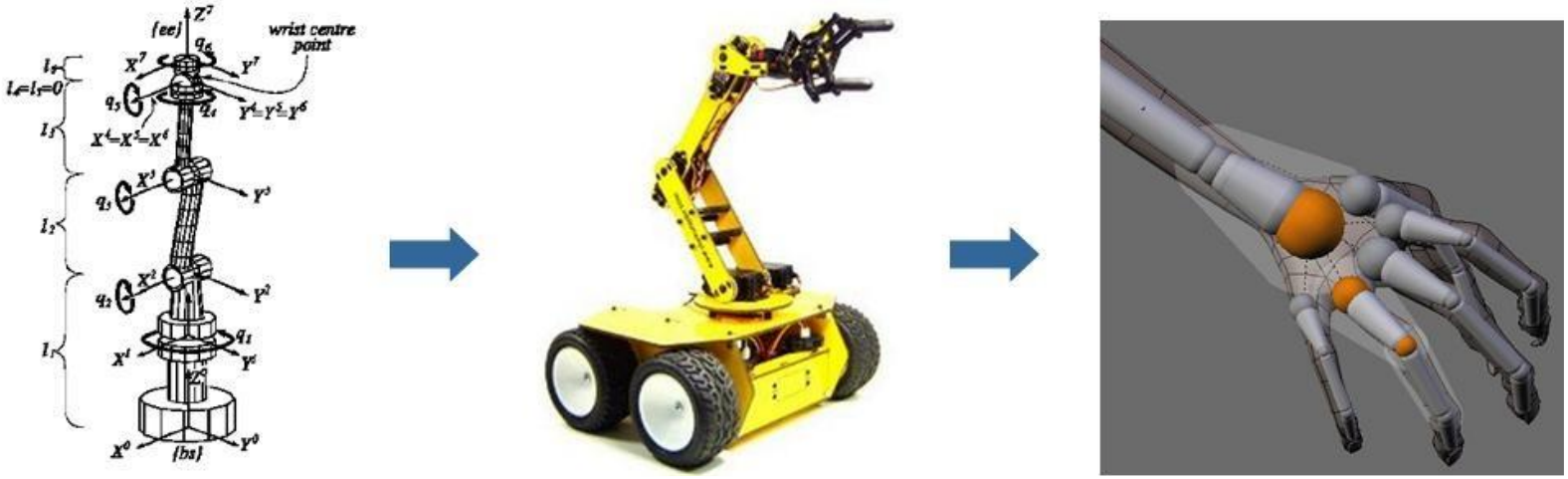
David Lanier, 2002

SQUELETTES D'ANIMATION

- Vocabulaire:
  - Un objet contenant une hiérarchie interne est appellé un **objet articulé**.
  - Le fait de modifier l'état des transformations de la hiérarchie s'appelle **l'articulation** d'un modèle. (À ne pas confondre avec "une articulation".)

SQUELETTES D'ANIMATION ET ROBOTIQUE

- L'idée d'utiliser un squelette pour modéliser le mouvement d'un modèle provient de la robotique.
  - La méthode a été ensuite adaptée à l'infographie et à l'animation par ordinateur.



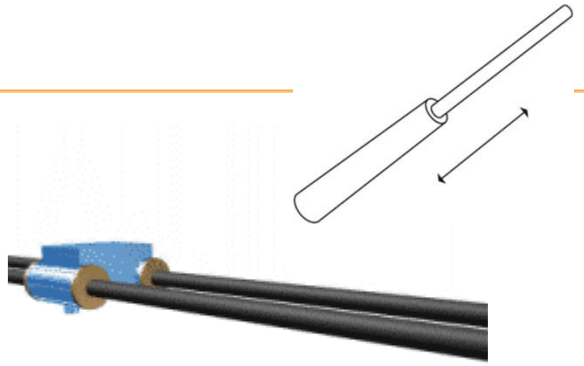
SQUELETTE D'ANIMATION ET ROBOTIQUE

- Un parallèle peut être établi entre le vocabulaire utilisé en robotique et celui utilisé pour décrire les squelettes d'animation.
  - En robotique, une série de constructions rigides reliées par des servo-moteurs s'appelle "**manipulateur**".
    - En animation on parle ici de **squelettes**.
  - Les dits moteurs utilisés en robotique sont appelés "**articulations**".
    - En animation on les nomme, "**articulations**" ou "**nœuds**".
  - Les constructions rigides sont appelés "**liens**" en robotique.
    - En animation, on les désigne comme étant des "**arcs**" ou des "**os**".

LES ARTICULATIONS

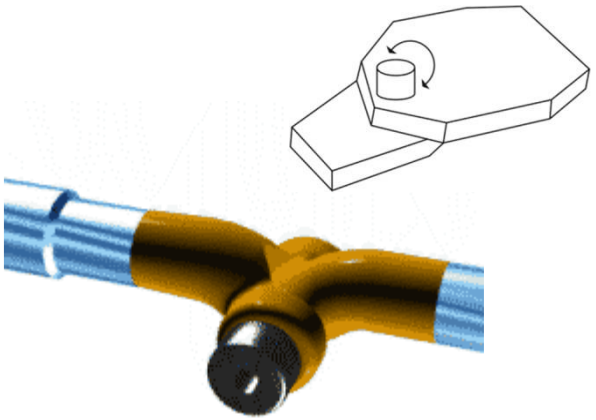
- Les articulations sont les points de jonction entre les différents arcs de notre squelette d'animation.

- Il existe deux types d'articulations primaires:
  - Les articulations prismatiques
    - Translation selon un axe quelconque.
  - Les articulations rotoïde
    - Rotation autour d'un axe quelconque.



Prismatique

University of Texas at Austin, RRG



Rotoïde

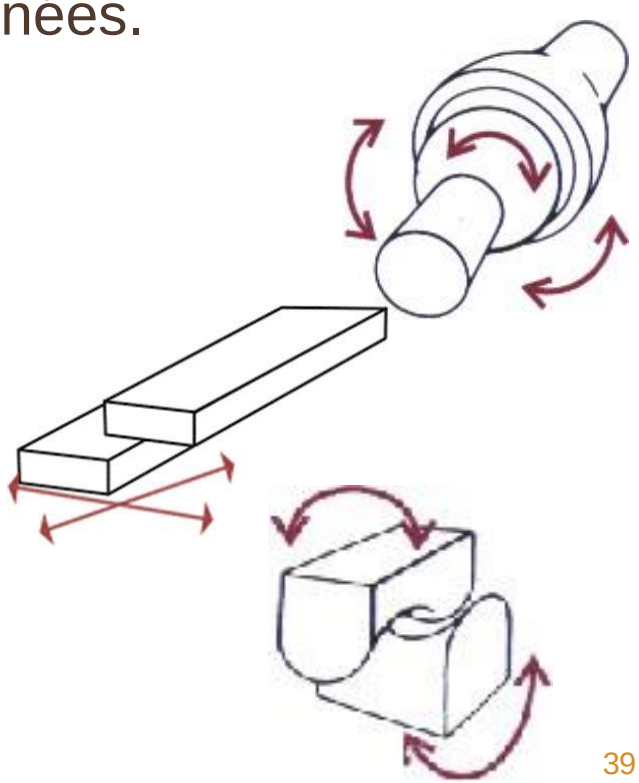
University of Texas at Austin, RRG

LES ARTICULATIONS

- Une des propriétés d'une articulation est le nombre de transformations distinctes qu'elle peut effectuer.
  - Cette propriété est appellée le **nombre de degrés de liberté**.
  - Les articulations primaires ont un seul degré de liberté.

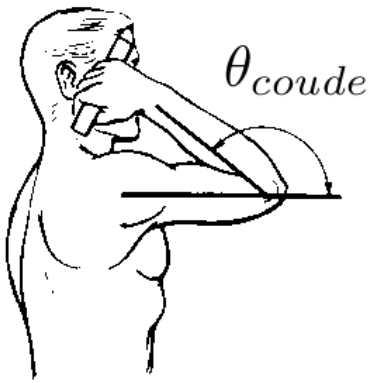
LES ARTICULATIONS

- Si une articulation possède plus d'un degré de liberté, on parle ici d'une **articulation complexe**.
  - Une articulation complexe peut être modélisée comme étant plusieurs articulations primaires combinées.
  - Exemples: (non exhaustif)
    - Articulation Rotule (Ball and socket)
      - Équivalent de 3 rotoïdes
        → 3 degrés de liberté.
    - Articulation Planaire (Planar)
      - Équivalent de 2 articulations prismatiques
        → 2 degrés de liberté
    - Articulation Selle (Saddle)
      - Équivalent de 2 rotoïdes orthogonaux
        → 2 degrés de libertés.



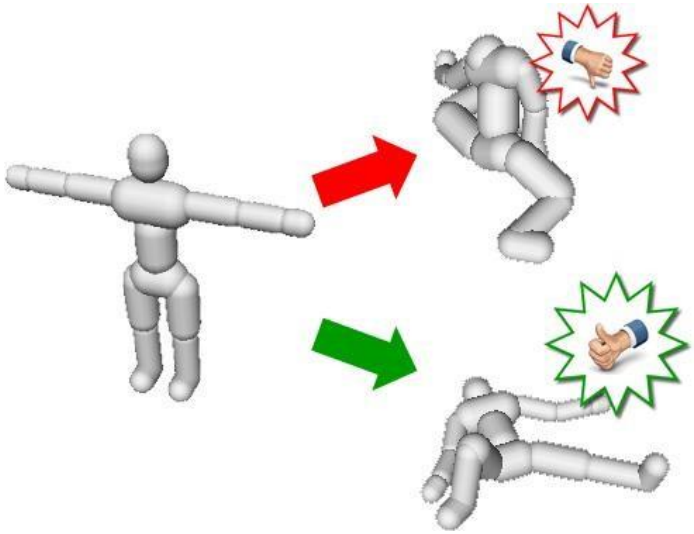
LES ARTICULATIONS

- Une articulation peut être complètement caractérisée par un nombre de paramètres équivalent au nombre de degrés de liberté qu'elle possède. Ces paramètres se nomment **variables d'articulation**.



$$\theta_{coude} = [0, 142^o]$$

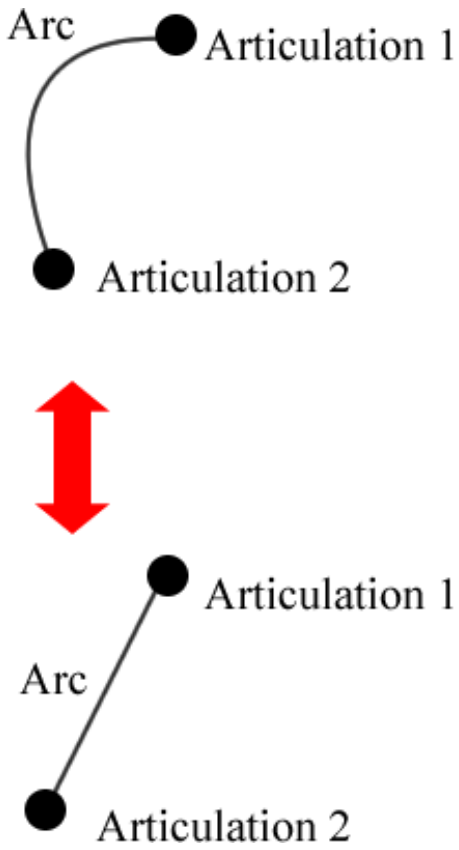
- En animation, il est courant de limiter les valeurs possibles des variables d'articulations. On parle ici de **contraintes d'articulation**.
  - Ceci permet de conserver l'intégrité et le réalisme visuel de notre modèle.
    → *Animation conventionnelle*
  - Ceci permet de conserver l'intégrité structurelle de notre modèle.
    → *Animation procédurale*



Monstrous Software, 2007

LES ARCS

- S'apparentent aux "os" du squelette.
  - Ils servent à définir la distance par défaut entre 2 articulations.
  - Un arc en soit est habituellement rigide dans le temps.
  - Un arc, est habituellement représenté par un prisme mais n'est en théorie qu'une distance.
    - Deux arcs représentés de façons différentes mais formant la même distance entre deux joints sont considérés identiques.



SQUELETTE D'ANIMATION

- Pourquoi utiliser le squelette d'animation?
  - Beaucoup plus simple à animer.
    - On manipule les articulations plutôt que d'avoir à manipuler chaque vertex indépendamment.
  - Beaucoup plus compact que l'animation par keyframe classique.
    - (Le modèle est stocké 1 fois en mémoire, on ne fait qu'animer le squelette.)
  - Assez simple pour permettre l'interaction avec l'environnement
  - Possibilité de réutiliser le même squelette pour plusieurs modèles 3D. (Mêmes animations, apparences différentes.)

SQUELETTE D'ANIMATION

- Désavantages d'un squelette d'animation:
  - La flexion au niveau des articulations peut être visible et causer des discontinuités dans le modèle.
    - La géométrie du modèle doit être construite en considérant le squelette.

MÉTHODES DE SKINNING

QU'EST-CE QUE LE SKINNING?

- Comme nous l'avons vu, le squelette d'animation est associé à un modèle.
  - Chaque point qui compose un modèle est associé à une ou plusieurs articulations.
  - Lorsqu'une articulation est transformée, les points qui y sont associés sont transformés en conséquence.
  - En assignant chacun des points du modèle à un squelette et en transformant (animant) celui-ci, on transforme et déforme le modèle, ce qui crée l'animation.

Microsoft, Bungie, 2005

QU'EST-CE QUE LE SKINNING?

- Le skinning se définit donc comme étant la méthode permettant :
  - D'assigner les points (vertex) qui composent un modèle aux différentes articulations de son squelette.
  - De transformer efficacement ces points au prorata des transformations du squelette.

LE SKINNING

- La notion de déformation d'objets 3D a été à la source même du développement des diverses méthodes de skinning. [*Magnenat-Thalmann et al. 1988*]

- Originalement, le skinning était surtout utilisé pour l'animation de personnages en 3D dans les jeux vidéos.
  - Les premiers algorithmes de skinning n'étaient habituellement pas publiés, le jeu vidéo n'étant alors pas un milieu très associé à la communauté scientifique.

LE SKINNING

- Il existe probablement autant de variantes aux algorithmes de skinning qu'il y a de chercheurs s'étant penché sur la question.

- Dans le cadre du cours, nous nous limiterons aux méthodes de bases les plus répandues, soit:

  - Assignation directe
  - Linear Blend Skinning (LBS)
  - Spherical Blend Skinning (SBS)

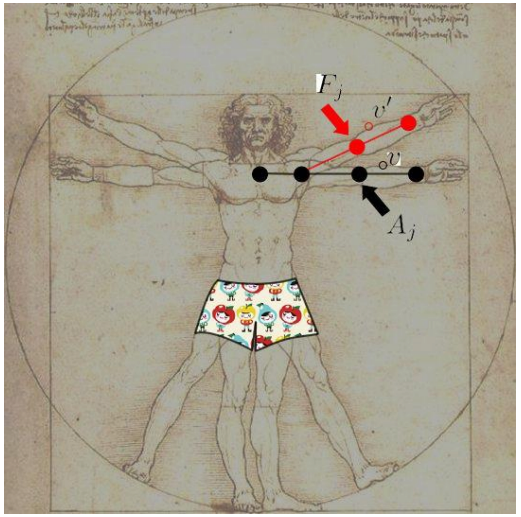
SKINNING PAR ASSIGNATION DIRECTE

- Le skinning par association directe est la première méthode de skinning a avoir été développée.

- Elle consiste simplement à assigner une articulation à un vertex, puis à transformer le vertex par la transformation exacte appliquée à l'articulation.

SKINNING PAR ASSIGNATION DIRECTE

- Prenons un modèle 3D possédant **n** articulations.
  - Chaque articulation est identifiée par un index **j**.
  - Le modèle 3D est dans sa position de référence (souvent la position de De Vinci).
  - La transformation d'une articulation j, dans sa position de référence, est caractérisée par la matrice de transformation $\mathbf{A}_j$.
  - Une même articulation **j**, une fois transformée par le mouvement du squelette, est caractérisée par la matrice de transformation $\mathbf{F}_j$.
  - Finalement, le vertex à transformer est dénoté **v**.

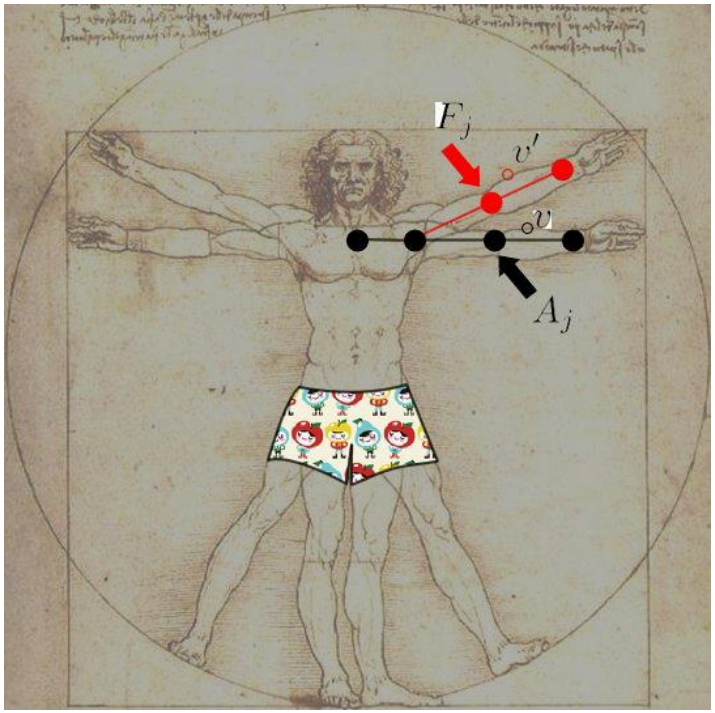


SKINNING PAR ASSIGNATION DIRECTE

- $\mathbf{A_j}$ peut être considéré comme la matrice de transformation de la position initiale.
- $\mathbf{F_j}$ la matrice de transformation pour transformer vers la position finale.
- Trouver la transformation de $\mathbf{A_j}$ à $\mathbf{F_j}$ revient donc à faire un changement de repère.

1. Il suffit d'enlever la transformation $\mathbf{A_j}$ avant d'appliquer $\mathbf{F_j}$

$$v' = F_j A_j^{-1} v$$



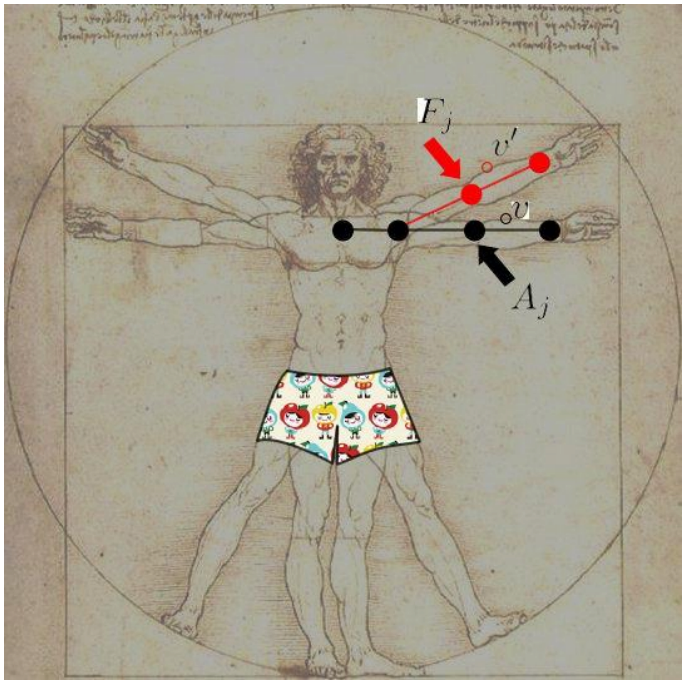
SKINNING PAR ASSIGNATION DIRECTE

- Donc :

$$v' = F_j A_j^{-1} v$$

- $\mathbf{F_j}$ et $\mathbf{A_j^{-1}}$ allant souvent de paire, on les exprime habituellement sous la forme:

$$C_j = F_j A_j^{-1}$$



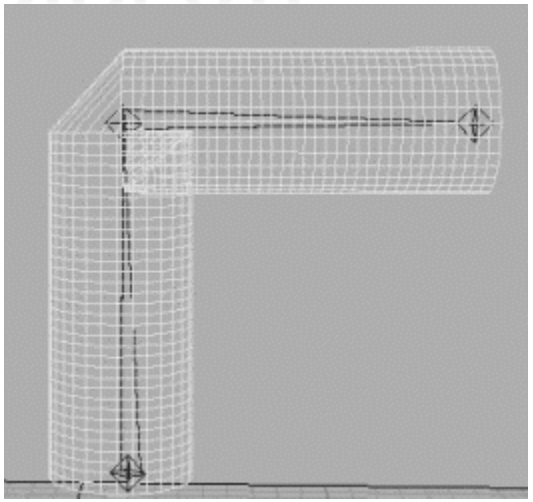
SKINNING PAR ASSIGNATION DIRECTE

- Somme toute, il est relativement simple de déformer un modèle pour qu'il suive un squelette:

1. On assigne une articulation à chaque vertex.
2. On transforme le vertex selon $v' = F_j A_j^{-1} v$ chaque fois que le squelette bouge.
3. On constate que le résultat est plutôt rudimentaire.

SKINNING PAR ASSIGNATION DIRECTE

- Lacunes:
  - Assigne un vertex à une articulation exclusivement.
  - Les vertex se situant aux limites d'une articulations se trouvent donc à replier le modèle sur lui-même ou à provoquer des déformations indésirées.
- Pour parer à ces lacunes, le Linear blend skinning a été développé.



David Lanier, 2002

Gareth Fouche, "Venture Dinosauria", 2007

LINEAR BLEND SKINNING

- Première alternative à l'assignation directe, encore une fois développée pour le jeu vidéo: **linear blend skinning (LBS)**. [*Lander 1998; Lander 1999*].

- Premier jeu à avoir utilisé et popularisé le linear blend skinning. Turok 2: Seeds of Evil.

Acclaim, 1998

LINEAR BLEND SKINNING

- Il existe plusieurs noms pour parler du linear blend skinning.

  + Skeleton subspace deformation
  + Vertex blending envelopping
  + Soft-Skinning, etc

  = Linear blend skinning

LINEAR BLEND SKINNING

- Globalement, le concept général du linear blend skinning est le suivant:
  - Plutôt que d'assigner 1 vertex à 1 seule articulation, on assigne 1 vertex à n articulations différentes.
  - La contribution de chaque articulation **j** à la transformation du vertex **v** est définie par un poids, noté $\mathbf{w}_j$.

LINEAR BLEND SKINNING

- Si on repart de l'assignation directe de vertex, on a:

$$v' = F_j A_j^{-1} v$$

- Et : $C_j = F_j A_j^{-1}$

- Ceci fonctionne bien si on souhaite assigner 1 vertex à 1 articulation. Pour assigner 1 vertex à n articulations, on utilise:

$$v' = \sum_{i=1}^{n} w_i C_{j_i} v$$

LINEAR BLEND SKINNING

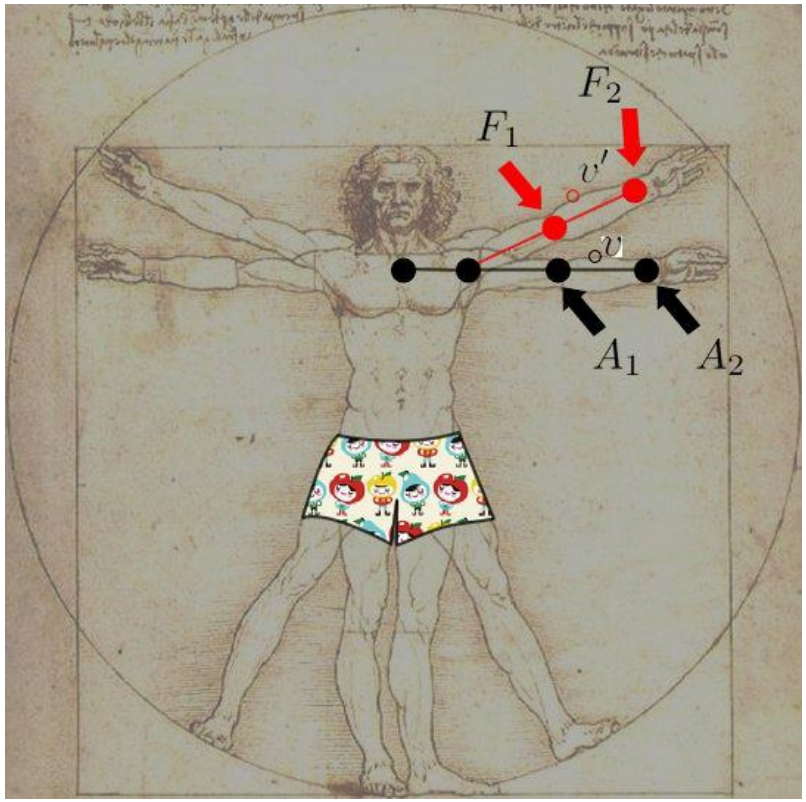
- Notons que pour : $v' = \sum_{i=1}^{n} w_i C_{j_i} v$
- On doit avoir : $\sum_{i=1}^{n} w_i = 1$
- Le tout revient donc à faire une somme pondérée de plusieurs transformations de vertex en fonction de multiples articulations.

LINEAR BLEND SKINNING

- Si on reprend notre exemple précédent, on aurait donc:

$$w_1 = 0.9$$
$$w_2 = 0.1$$

$$v' = \sum_{i=1}^{2} w_i C_{j_i} v$$
$$= w_1 F_1 A_1^{-1} v + w_2 F_2 A_2^{-1} v$$
$$= (0.9) F_1 A_1^{-1} v + (0.1) F_2 A_2^{-1} v$$



LINEAR BLEND SKINNING

- Pourquoi effectuer :

$$v' = \sum_{i=1}^{n} w_i C_{j_i} v$$

plutôt que:

$$v' = (\sum_{i=1}^{n} w_i C_{j_i}) v \quad ?$$

- Souvenons-nous au début de la session, lorsqu'on parlait d'interpoler entre matrices de transformation:

$$R_{y_0}(90^o) = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ -1 & 0 & 0 \end{bmatrix} \quad R_{y_1}(-90^o) = \begin{bmatrix} 0 & 0 & -1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix} \quad (0.5)R_{y_0} + (0.5)R_{y_1} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

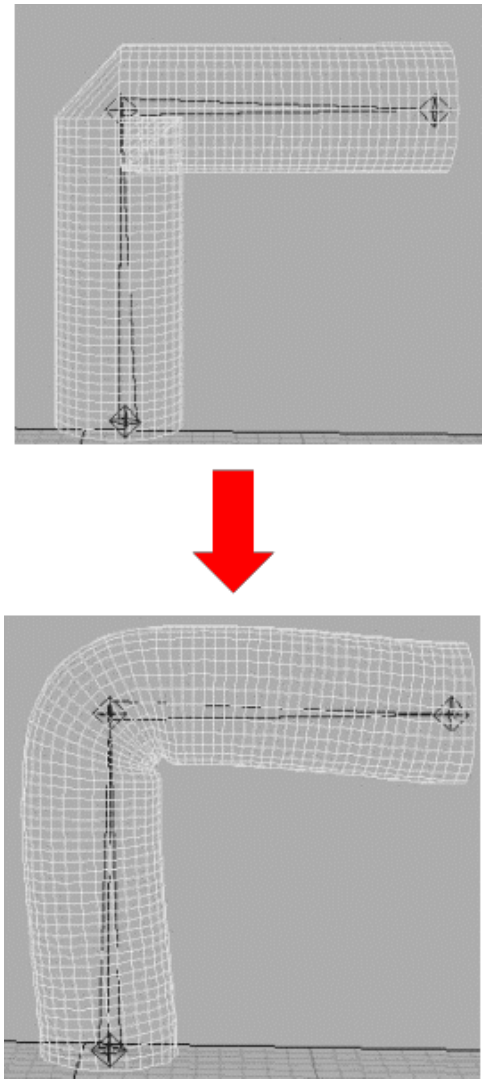
- On veut éviter de se retrouver avec une matrice singulière pour transformer le vecteur.

LINEAR BLEND SKINNING

- Comment détermine-t-on w_i?
  - Souvent déterminé manuellement par un artiste.
    - Il existe des outils spécialisés pour ce genre de tâches reliées au skinning, nous les verrons plus tard.
  - Sinon: généré automatiquement avec des heuristiques (par exemples les 4 articulations les plus proches sont associés au vertex au prorata de leur distance.)

LINEAR BLEND SKINNING

- Avantages du LBS
  - Conserve mieux l'intégrité du modèle 3D
    - Évite les cassures et les étirement extrême au niveau des articulations.
    - Permet de tenir compte des différentes transformations locales du modèle 3D plutôt que celle d'une seule articulation.
  - S'exécute très rapidement et ne requiert qu'une série de transformations géométriques simples.
    - Peut facilement être implémenté en tant que shader sur le processeur graphique.



David Lanier, 2002

LINEAR BLEND SKINNING

- Le linear blend skinning crée néanmoins certains artefacts de rendu peu désirables [*Steed 2000*]:
  - Perte de volume au niveau des articulations.
  - Effet dit “d’emballage de bonbon” (candy wrap) lorsque les articulations sont tournées sur l’axe de l’arc leur correspondant.

Michiel van de Panne, 2006

[Jacka, Reid et al.,2007]

SPHERICAL BLEND SKINNING

- Pour remédier aux problèmes du LBS, divers correctifs ont été proposés:
  - LBS pour interpoler entre des états pré-générés du skin. [*Sloan et al. 2001*]
  - Correction des artefact utilisant une division par composantes principales du modèle [*Kry et al. 2002*].
- L'alternative la plus répandue et utilisée est probablement **Spherical Blend Skinning** [*Kavan,Zara,2005*] et ses variantes.

SPHERICAL BLEND SKINNING

- Plutôt que de tenter de corriger le Linear blend skinning, le Spherical blend skinning se démarque en utilisant une approche différente pour combiner les diverses transformations d’articulation.

SPHERICAL BLEND SKINNING

- Pour n articulations, on sait qu'on a n matrices de transformations définies telles que:

$$C_{j_i} = F_{j_i} A_{j_i}^{-1}$$

- On cherche ici à combiner les matrices $\mathbf{C_{ji}}$, comme pour le LBS, mais cette fois sans utiliser une interpolation linéaire directe pour combiner les transformations. On utilise plutôt:
  - Une combinaison linéaire des vecteurs de translation des articulations.
  - Une combinaison linéaire pour combiner les rotations des articulations **converties en quaternions**.

SPHERICAL BLEND SKINNING

- Nous avons donc une série de matrices de transformation telle que $C_{j_1}..C_{j_n}$ qu'on souhaite combiner au prorata de poids $w_1..w_n$ (qu'on peut exprimer comme étant éléments du vecteur **W**).

- L'opération utilisée pour la combinaison des matrices selon le poids est notée: $q(W; C_{j_1}, ..., C_{j_n})$

- Puisque la translation et la rotation sont combinées séparément, on les extrait de chaque matrice de façon à les séparer:
  - La translation extraite d'une matrice C_{ji} est notée: $C_{j_i}^{tr}$
  - La rotation elle est extraite de $\mathbf{C_{ji}}$ (matrice 3x3 supérieure gauche) et convertie en quaternion.

    On note le quaternion : q_{j_i}

SPHERICAL BLEND SKINNING

- La combinaison des vecteurs de translation nous donne un vecteur noté **m** et s'effectue linéairement au prorata du poids (moyenne pondérée), soit:

$$m = \sum_{i=1}^{n} w_i C_{j_i}^{tr}$$

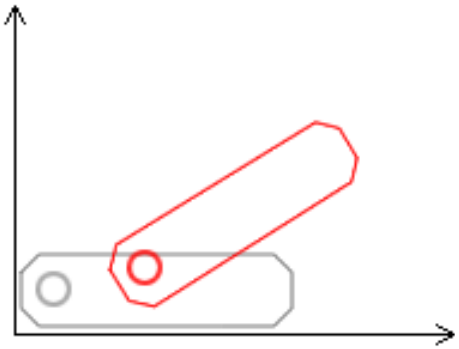
- La combinaison des quaternions nous donne un quaternion noté $\mathbf{s_n}$ et s'effectue en utilisant une moyenne pondérée de façon analogue :

$$\begin{aligned} s &= w_1 q_{j_1} + \ldots + w_n q_{j_n} \\ &= \sum_{i=1}^{n} w_i q_{j_i} \\ s_n &= s/||s|| \end{aligned}$$

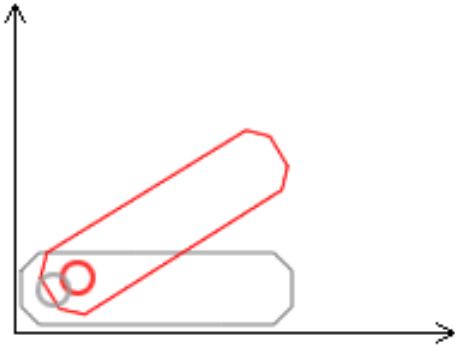
- Une fois $\mathbf{s_n}$ trouvé, on converti $\mathbf{s_n}$ en matrice de rotation notée **Q.**

SPHERICAL BLEND SKINNING

- Une fois **Q** et **m** trouvés, on peut reconstruire une matrice de rotation correspondant à la combinaison de toutes nos matrices $\mathbf{C_{ji}}$.
- On a donc: $q(W; C_{j_1}, ..., C_{j_n}) = \begin{pmatrix} Q_{(3\times3)} & m \\ 0 & 1 \end{pmatrix}$
- Il reste un dernier problème :
  - En effectuant cette transformation, on fixe un centre de rotation arbitraire qui n'est pas optimal. En résulte un certain décalage spatial des points.
  - On doit donc trouver un point dans l'espace qui servira de centre de rotation et qui réduit au maximum le dit décalage.



Mauvais centre de rotation



Meilleur centre de rotation

SPHERICAL BLEND SKINNING

- Notre but est donc de trouver un point affecté de façon égale par chacune des transformations d'articulations.

- Pour 2 articulations **a** et **b**, on cherche donc un point $\mathbf{r_c}$ (pour "rotation center") qui réalise l'égalité:

$$C_a r_c = C_b r_c$$

- Pour **n** articulations, on cherche un $\mathbf{r_c}$ tel que toute combinaison de 2 de ces **n** articulations préserve l'égalité. Soit:

$$C_a r_c = C_b r_c, \;\; a < b, \quad a, b \, \epsilon \, (j_1, .., j_n)$$

SPHERICAL BLEND SKINNING

- Or, on sait que $\mathbf{C_j}$ est défini tel que:

$$C_j = \begin{pmatrix} C_j^{rot} & C_j^{tr} \\ 0 & 1 \end{pmatrix}$$

- On peut donc exprimer l'égalité précédente tel que:

$$\begin{aligned} C_a r_c &= C_b r_c \\ C_a^{rot} r_c + C_a^{tr} &= C_b^{rot} r_c + C_b^{tr} \end{aligned}$$

- En manipulant un peu la formule précédente, on obtient:

$$(C_a^{rot} - C_b^{rot}) r_c = C_b^{tr} - C_a^{tr}$$

SPHERICAL BLEND SKINNING

- Or, comme on l'a indiqué précédemment, cette égalité doit être vraie pour tout duo de matrices impliquée dans la combinaison de transformations des articulations, donc:

$$(C_a^{rot} - C_b^{rot})r_c = C_b^{tr} - C_a^{tr},\ a < b,\quad a, b\ \epsilon(j_1, .., j_n)$$

- On peut exprimer cette série d'égalités (toutes les combinaisons possibles) en un seul gros système linéaire:

$$Dr_c = e$$

- Où **D** correspond à toutes les combinaisons $(C_a^{rot} - C_b^{rot})$ possibles.

- Et **e** à toutes les combinaisons $(C_b^{tr} - C_a^{tr})$ possibles.

SPHERICAL BLEND SKINNING

- Malheureusement, la matrice D est difficilement inversible et le système peut paraître irrésolvable:

$$\begin{aligned} D &\rightarrow 3\binom{n}{2} \times 3 \\ r_c &\rightarrow 3 \times 1 \\ e &\rightarrow 3\binom{n}{2} \times 1 \end{aligned}$$

- Notre système est sur-contraint et à moins d'un cas particulier, il n'existe pas de solution $\mathbf{r_c}$ rendant l'égalité possible pour toutes les combinaisons de transformation **a** et **b**.

SPHERICAL BLEND SKINNING

- Notre objectif n'est donc pas de trouver un $\mathbf{r_c}$ éliminant le décalage mais de trouver le $\mathbf{r_c}$ causant le moins grand décalage. (en terme de distance euclidienne)

- Comment trouver un tel $\mathbf{r_c}$?
  - Solution → Décomposition de **D** en valeurs singulières (dvs).
  - La solution calculée par la dvs d'un système sans solution de forme Ax = b correspond à la solution des moindres carrés.
  - Pour calculer la décomposition en valeurs singulières utiliser une librairie d'algèbre linéaire (comme LAPACK par exemple.)

SPHERICAL BLEND SKINNING

- On a maintenant trouvé notre centre de rotation $\mathbf{r_c}$ provoquant le moins de décalage possible.
- Pour que soit effectif le choix du nouveau centre de rotation, on crée une matrice de transformation servant à recentrer nos transformations autour de $\mathbf{r_c}$, soit:

$$T = \begin{pmatrix} I & r_c \\ 0 & 1 \end{pmatrix}$$

- On prend ensuite notre combinaison de matrices initiales:

$$q(W; C_{j_1}, ..., C_{j_n})$$

- Qu'on centre autour de $\mathbf{r_c}$:

$$Tq(W; T^{-1}C_{j_1}T, ..., T^{-1}C_{j_n}T)T^{-1}$$

SPHERICAL BLEND SKINNING

- Une fois la transformation finale obtenue, on peut transformer rapidement nos vertex : (ici on a un vertex **v**)

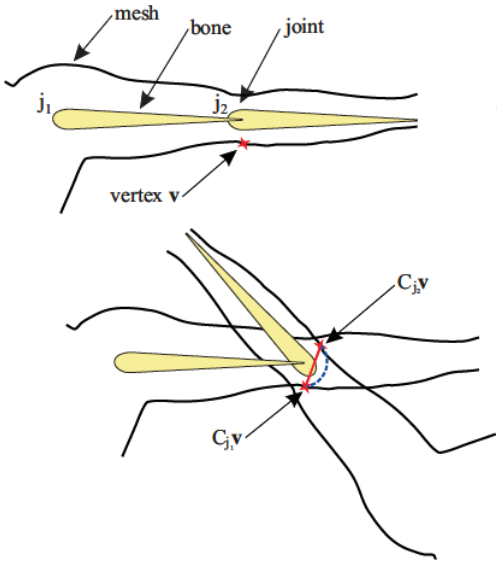
$$v' = Tq(W; T^{-1}C_{j_1}T, ..., T^{-1}C_{j_n}T)T^{-1}v$$

SPHERICAL BLEND SKINNING

- En résumé:

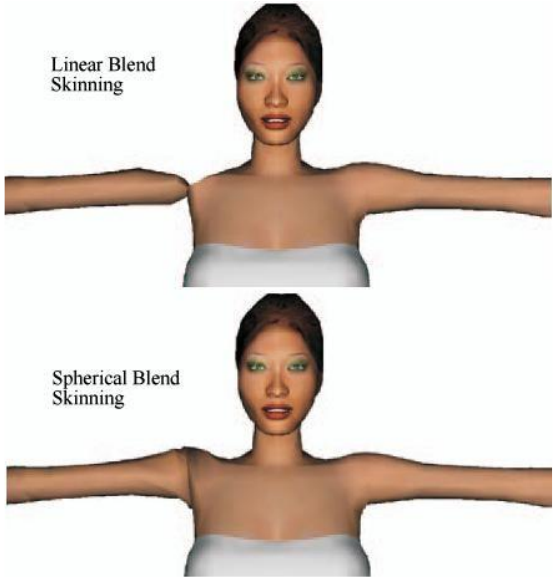
1. Séparer la transformation de translation et de rotation dans les différentes matrices de transformation.
    1. Utiliser des quaternions pour représenter l'orientation
    2. Utiliser un vecteur pour représenter la translation.
2. Combiner les quaternions au prorata du poids de chaque articulation.
3. Faire la moyenne pondérée des différentes translations.
4. Trouver un point à utiliser comme centre de rotation avec la décomposition en valeurs singulières.
5. Prendre la combinaison des quaternions et la moyenne des translations puis appliquer la transformation à partir du centre de rotation trouvé en (4).

SPHERICAL BLEND SKINNING



Kavan Zara, 2005

- Avantages:
  - Élimine la plupart des problèmes de perte de volume.
  - Réduit/élimine la plupart des problèmes de déformation style “emballage de bonbons” (candy wrap).
- Désavantages:
  - Beaucoup plus complexe à comprendre et implémenter (quaternions, décomposition par valeurs singulières, etc.)
  - Plus gourmand en temps de calcul.



Kavan Zara, 2005

TECHNIQUES DE SKINNING

- Quelle est la meilleure technique?
  - Chaque technique possède ses forces, selon le cas, une technique sera plus avantageuse ou mieux adaptée qu'une autre.
  - Exemple:
    - Personnage dont les pièces sont rigides (Robot, armure, etc.)
      → **L'assignation directe** de vertex est préférable.
    - Séquence d'animation pré-rendue, moteurs 3D de pointe
      → **Spherical blend skinning** et modifieurs d'influence (que nous verrons à la prochaine section).
    - Quantité élevée d'animations simples et simultanées dans un jeu.
      → **Linear blend skinning**.

De haut en bas: Pixar "Wall-e", 2008, ILM "Shrek", 2001, Sega "Rome Total War ", 2001

OUTILS UTILISÉS POUR LE SKINNING

OUTILS UTILISÉS POUR LE SKINNING

- Le skinning peut être une tâche longue et ennuyeuse si elle n'est pas assistée par les bons outils.

- Sans outils, elle consisterait à assigner une série d'articulations manuellement à chaque vertex, en définissant manuellement et individuellement chacun des poids.

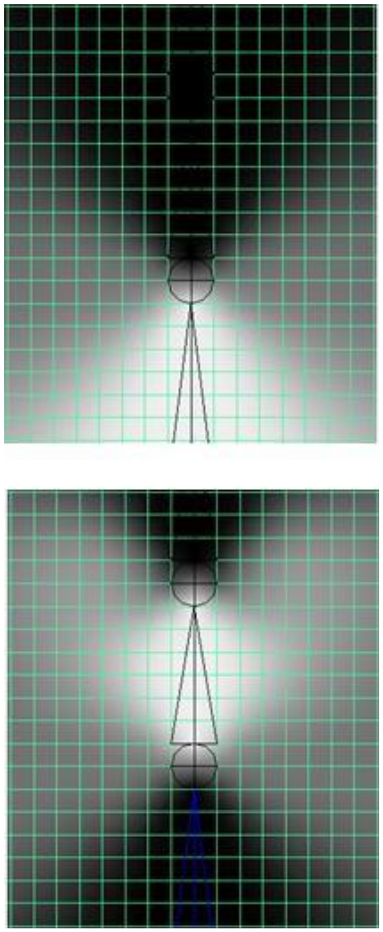
  - Peu intuitif et long à construire
  - Beaucoup trop sujet à erreur.

OUTILS UTILISÉS POUR LE SKINNING

- Les algorithmes d'assignation complètement automatisés sont habituellement imprécis ou tendent à regrouper des parties du squelette qui ne devraient pas l'être.
- Diverses alternatives ont donc été développées pour faciliter le travail des artistes.
- Habituellement, trois outils sont utilisés dans le processus de skinning:
  - Assignation automatique supervisée.
  - Peinture de cartes de poids (weight maps)
  - Objets d'influence (Influence objects)

OUTILS UTILISÉS POUR LE SKINNING

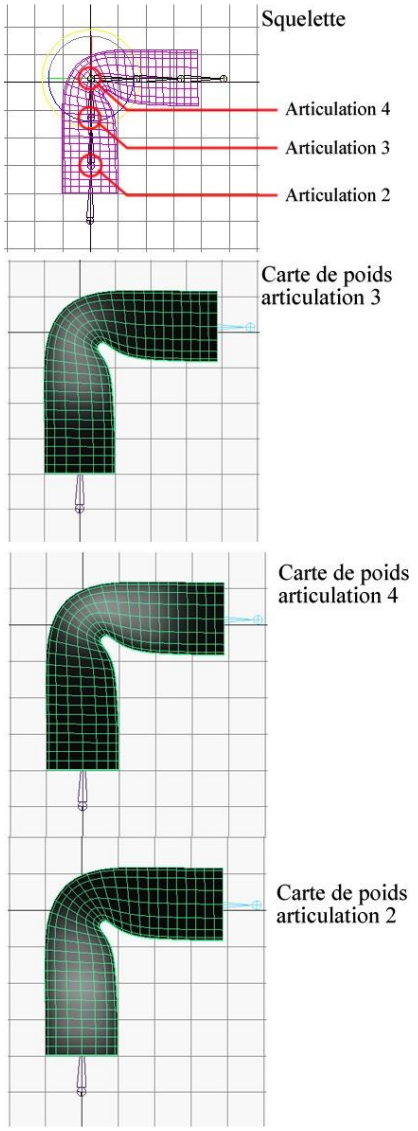
- Assignation automatique supervisée.
  - Assigne automatiquement les vertex sélectionnés à des articulations sélectionnées.
    - Permet à l'utilisateur de déterminer quels groupes de vertices sont associés à quels groupes d'articulations. (Plutôt que de laisser l'ordinateur tout décider de A à Z.)
    - L'assignation des poids en soit peut se fait de différentes façons. Elle est cependant habituellement faite en utilisant la distance du vertex par rapport aux articulations et leurs os.



Assignation automatique des poids des vertices.

OUTILS UTILISÉS POUR LE SKINNING

- Cartes de poids (Weight map)
  - Une méthode efficace pour représenter le poids de chacun des vertices par rapport à une articulation est d'utiliser une carte colorée du poids des vertices par rapport à une articulation.
  - La carte de poids permet de visualiser quel est le poids d'une articulation pour les différents vertices du modèles.
  - Dans l'exemple à droite, plus un vertices est affecté par l'articulation, plus il est pâle.

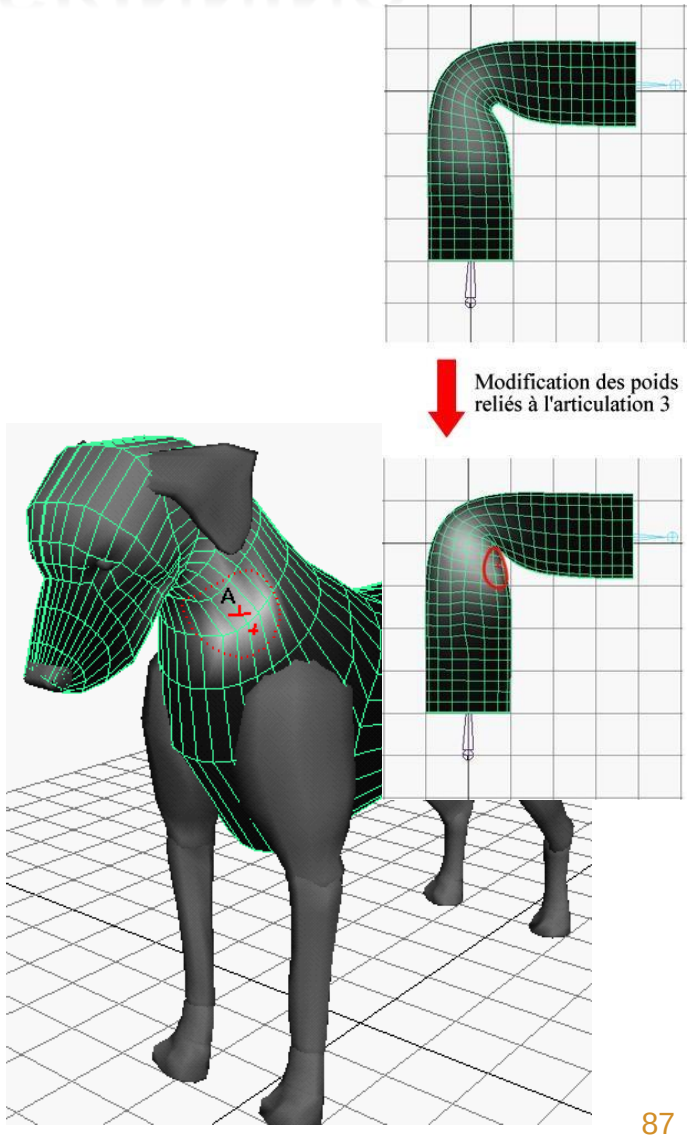


OUTILS UTILISÉS POUR LE SKINNING

- Peinture de carte de poids (Weight map painting)
  - L'assignation automatique supervisée des poids est habituellement la première étape du skinning. Elle produit une assignation de départ relativement correcte entre les vertices et les articulations.
  - Cette assignation, bien que complète, reste imparfaite. On peut la considérer comme une ébauche de la carte de poids final.
  - En effet, la carte de poids générée de cette façon correspond rarement parfaitement à celle désirée.

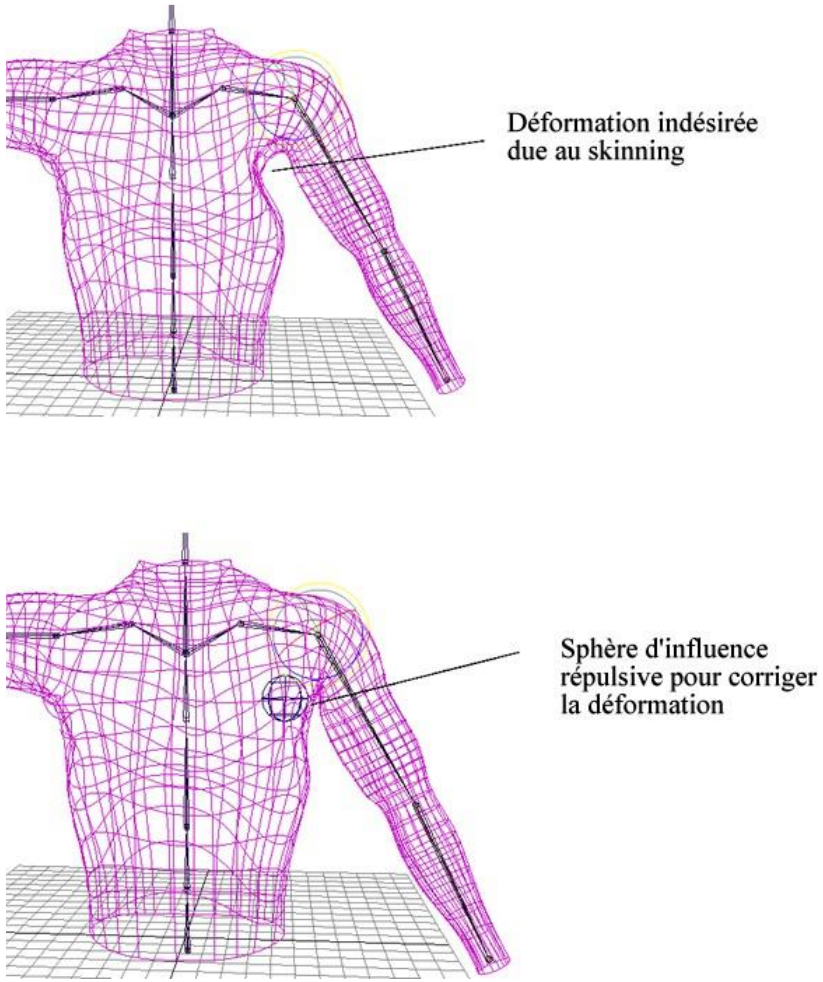
OUTILS UTILISÉS POUR LE SKINNING

- Peinture de carte de poids (Weight map painting)
  - Pour régler un tel problème, on utilise une méthode appellée “peinture de carte de poids”.
  - Cette méthode revient à dessiner directement le poids de chacun des vertices sur le modèle 3D, pour une articulation donnée.
  - L’opération de peinture s’effectue de la même façon et avec des outils similaires à ceux qu’on retrouve dans photoshop ou paint, à l’exception qu’elle se fait sur la surface d’un modèle 3D.
  - On obtient ainsi une assignation des poids plus précise et on peut régler les problèmes localisés résultants de l’assignation automatique supervisée.



OUTILS UTILISÉS POUR LE SKINNING

- Objets d'influence (Influence objects)
  - Dernière étape pour raffiner le skinning effectué.
  - Consiste à placer des formes géométriques qui vont servir d'attracteurs ou de répulseurs aux vertices situés à proximité.
  - Ceci permet de compenser des déformations habituellement observées avec le squelette.
  - Les formes en question sont habituellement des primitives simples:
    - Sphères, coniques, quadriques



OUTILS UTILISÉS POUR LE SKINNING

- Sommaire:
  - On débute habituellement le skinning avec une **méthode d'assignation automatique supervisée**. (Pour faire le gros du travail.)
  - On ajuste en détail les poids pour corriger les erreurs induites par la méthode automatique en **peinturant notre carte de poids**.
  - On enlève les déformations indésirées restante ne pouvant être enlevées autrement à l'aide d' **objets d'influence.**

Annexe au chapitre 4

LANGAGES D'ANIMATION (MANUEL, 4.2)

LANGAGES D'ANIMATION

- Définition:
  - Un langage d'animation est un regroupement de structures de contrôle servantà encoder l'information nécessaire à la création d'une animation.

- Il existe différents types de langage d'animation:

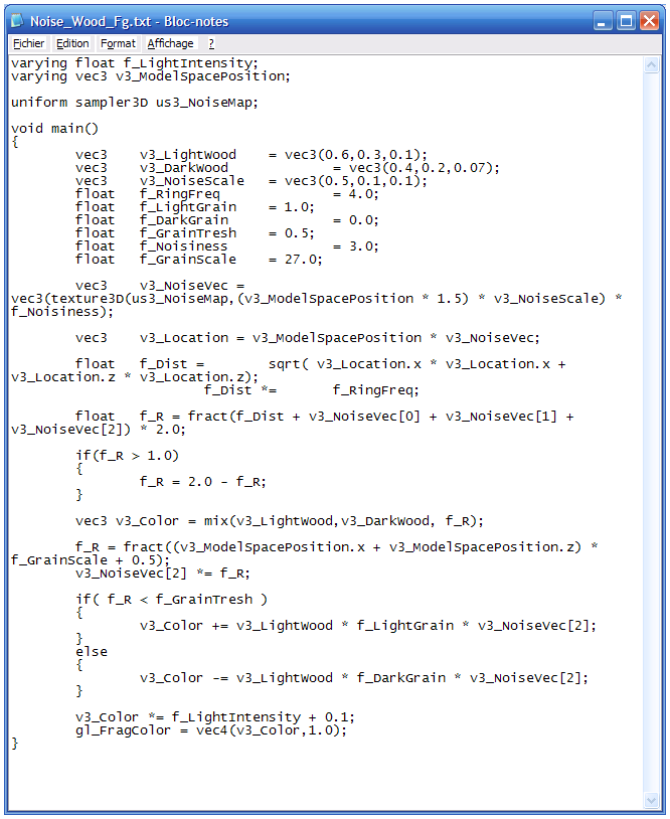
  - Langages de programmation complets.
  - Langages de programmation orientés artiste.
  - Langages centrés variables d'articulation.
  - Langages graphiques
  - Langages centrés acteurs

LANGAGES DE PROGRAMMATION COMPLETS

- Les premiers langages d'animation étaient presque tous des langages de programmation ou d'assembleur standards (Fortran, C, SPARC, etc.) utilisés pour générer une application produisant le rendu et l'animation.

- Chaque fois qu'une animation était produite, du code devait être écrit pour générer les primitives, les envoyer au matériel graphique, spécifier la méthode de rendu, etc.

- C'est un processus long et laborieux qui demeure peu productif.


Noise_Wood_Fg.txt - Bloc-notes

Fichier Edition Format Affichage ?

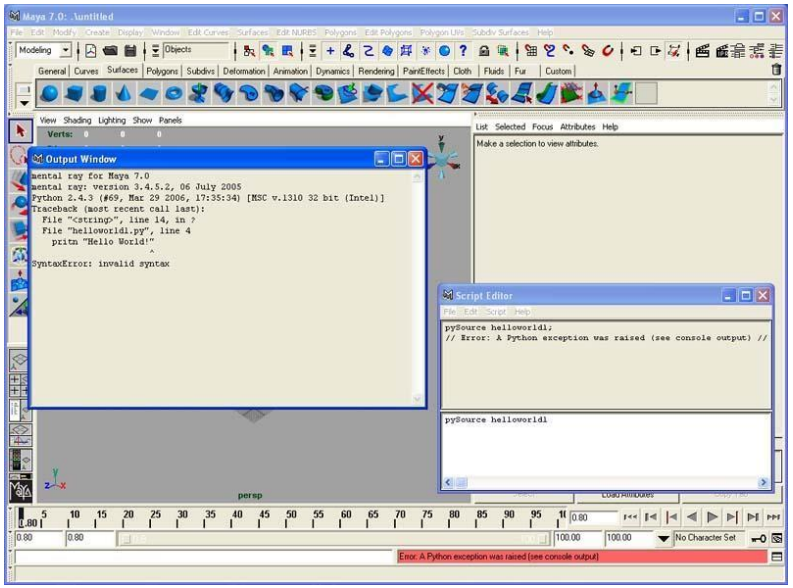
```
varying float f_LightIntensity;
varying vec3 v3_ModelSpacePosition;

uniform sampler3D us3_NoiseMap;

void main()
{
        vec3    v3_LightWood    = vec3(0.6,0.3,0.1);
        vec3    v3_DarkWood             = vec3(0.4,0.2,0.07);
        vec3    v3_NoiseScale   = vec3(0.5,0.1,0.1);
        float   f_RingFreq              = 4.0;
        float   f_LightGrain    = 1.0;
        float   f_DarkGrain             = 0.0;
        float   f_GrainTresh    = 0.5;
        float   f_Noisiness             = 3.0;
        float   f_GrainScale    = 27.0;

        vec3    v3_NoiseVec =
vec3(texture3D(us3_NoiseMap,(v3_ModelSpacePosition * 1.5) * v3_NoiseScale) *
f_Noisiness);

        vec3    v3_Location = v3_ModelSpacePosition * v3_NoiseVec;

        float   f_Dist =        sqrt( v3_Location.x * v3_Location.x +
v3_Location.z * v3_Location.z);
                        f_Dist *=       f_RingFreq;

        float   f_R = fract(f_Dist + v3_NoiseVec[0] + v3_NoiseVec[1] +
v3_NoiseVec[2]) * 2.0;

        if(f_R > 1.0)
        {
                f_R = 2.0 - f_R;
        }

        vec3 v3_Color = mix(v3_LightWood,v3_DarkWood, f_R);

        f_R = fract((v3_ModelSpacePosition.x + v3_ModelSpacePosition.z) *
f_GrainScale + 0.5);
        v3_NoiseVec[2] *= f_R;

        if( f_R < f_GrainTresh )
        {
                v3_Color += v3_LightWood * f_LightGrain * v3_NoiseVec[2];
        }
        else
        {
                v3_Color -= v3_LightWood * f_DarkGrain * v3_NoiseVec[2];
        }

        v3_Color *= f_LightIntensity + 0.1;
        gl_FragColor = vec4(v3_Color,1.0);
}
```

LANGAGES DE PROGRAMMATION COMPLETS

- Pour accélérer le processus, on a mis au point différentes librairies consacrées exclusivement au rendu 3D:
  - OpenGL, DirectX, OGRE, etc.
  - Ces librairies ont permis une définition simplifiée des éléments impliqués dans le rendu, ce qui facilite la création d'animation de façon indirecte.

LANGAGES DE PROGRAMMATION COMPLETS

- Cette évolution a mené vers une tendance:
  - Construire des applications d'animation utilisant une interface graphique mais fonctionnant en arrière plan avec un langage de script qui s'apparente à un langage de programmation complet.
  - Exemple: Maya et Mel script



Autodesk, Maya 7.0

LANGAGES D'ANIMATION ORIENTÉS ARTISTE

- Dans le but de rendre la programmation le plus accessible possible aux artistes, certains langages dit "orientés artiste" ont vu le jour.

- Ils se définissent par:

  - Des fonctions standards au langage expressément faites pour l'animation. (Interpolation, courbe, etc.)

  - Des structures et des syntaxes simplifiées. (Langages non typés, systèmes de noms simples plutôt que des variables, syntaxe s'apparentant à l'écriture.)

  - Une syntaxe fonctionnant par commandes.
    → 1 ligne = 1 action

LANGAGES D'ANIMATION ORIENTÉS ARTISTE

- Exemple d'un tel langage: ANIMA II

```
set position <name> <x> <y> <z> at frame  <number>

set rotation <name> [X,Y,Z] to <angle> at frame  <number>

change position <name> to <x> <y> <z> from frame <number> to  frame<number>

change rotation <name> [X,Y,Z] by <angle> from frame <number> to  frame<number>
```

LANGAGES D'ANIMATION ORIENTÉS ARTISTE

- Cependant:
  - La simplification de ces langages les rend plus facile à comprendre mais beaucoup moins performants à cause de cette dite simplification:
    - Suppression des boucles
    - Pas de notion d'objets ou de classe
    - Etc.
  - Un artiste devenant suffisamment familier avec un tel langage voudra habituellement qu'on y ajoute des fonctionnalités pour le rendre plus puissant:
    - Boucles
    - Objects/classe
    - Etc.

LANGAGES D'ANIMATION ORIENTÉS ARTISTE

- Pour pallier à ces lacunes, des langages complets sont développés. Ces derniers intégrent néanmoins toutes les fonctionnalités requises pour l'animation à l'intérieur même de celui-ci.

- L'objectif est habituellement de simplifier le langage sans pour autant couper dans ses fonctionnalités.

- Souvent accompagnés d'un IDE simple et intuitif.

LANGAGES D'ANIMATION ORIENTÉS ARTISTE

- Exemple d'un tel langage: Processing 1.0
  - Utilise la syntaxe de java.
  - Contient par défaut:
    - Librairie de formes (ellipse, cercle, carré, étoile, etc.)
    - Librairie de couleurs
    - Librairie de gestion d'images
    - Librairie de math appliquées (interpolation, distances, etc.)
    - Librairie de typographie
    - Etc.
  - IDE complètement intégré, aucune configuration nécessaire.
  - Simplification des structures d'application:
    - Fonction setup ← Chargement des objets, définition de variables.
    - Fonction draw ← Appellé chaque fois que doit se faire le rendu.

LANGAGES D'ANIMATION ORIENTÉS ARTISTE

- Un langage comme processing reste donc à la fois simple:
  - Séparé en petites librairies simples
  - Structure d'application simplifiée
- Mais peu atteindre une certaine complexité selon les besoins de l'utilisateur:
  - Utilise java donc orienté objet
  - Permet l'importation de librairies externes

LANGAGES D'ANIMATION ORIENTÉS ARTISTE

- Processing : Exemple simple

```
float r;

// Angle and angular velocity, accleration
float theta;
float theta_vel;
float theta_acc;

void setup() {
  size(200,200);
  frameRate(30);
  smooth();

  // Initialize all values
  r = 50.0f;
  theta = 0.0f;
  theta_vel = 0.0f;
  theta_acc = 0.0001f;
}

void draw() {
  background(0);
  // Translate the origin point to the center of the screen
  translate(width/2,height/2);

  // Convert polar to cartesian
  float x = r * cos(theta);
  float y = r * sin(theta);

  // Draw the ellipse at the cartesian coordinate
  ellipseMode(CENTER);
  noStroke();
  fill(200);
  ellipse(x,y,16,16);

  // Apply acceleration and velocity to angle (r remains static in this example)
  theta_vel += theta_acc;
  theta += theta_vel;

}
```

-Crée une fenêtre
-Ajuste le nombre d'images par seconde.
-Affiche un cercle qui tourne en suivant un trajet d'ellipse à vitesse constante.

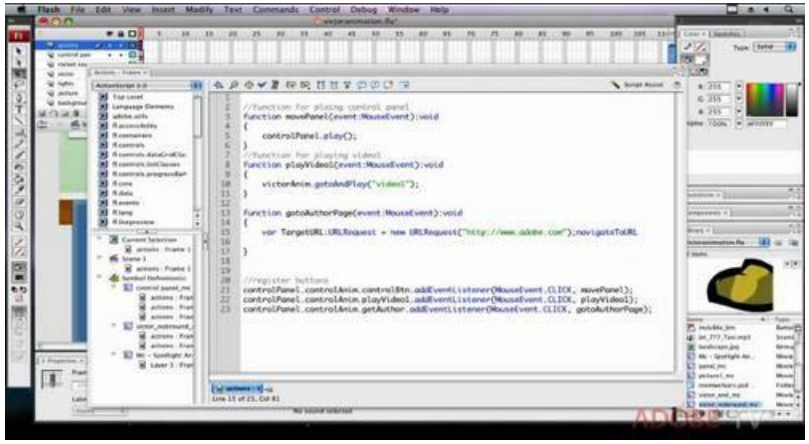
L'utilisateur n'a pas à se soucier du temps, de la gestion des fenêtres, de la création des formes, de la structure de l'application, etc.

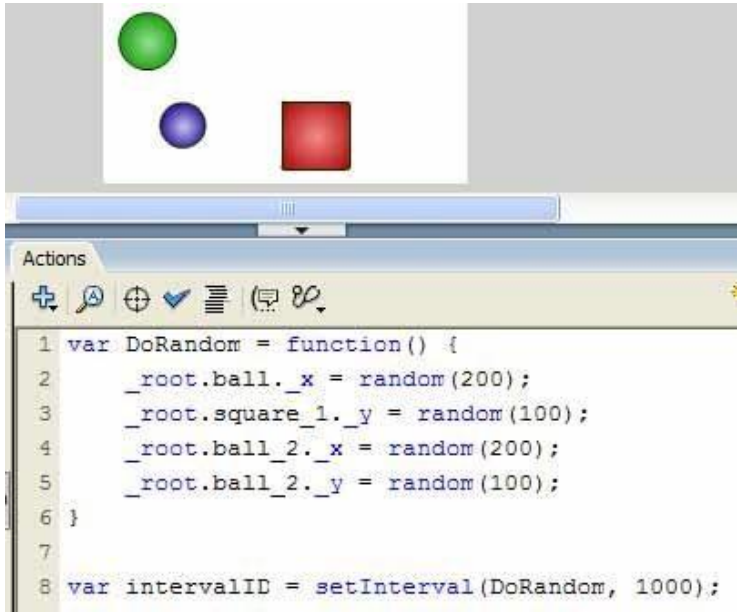
LANGAGES D'ANIMATION ORIENTÉS ARTISTES

- Les langages d'animation orientés artistes possèdent quand même quelques défauts:
  - Simplification des fonctionnalités → couche d'encapsulation supplémentaire
    - Performance moins élevées.
    - Moins adapté pour l'animation en temps réel.
    - Aucun accès rapide et direct au matériel (hardware).
  - La simplification des structures d'application et les IDE plus simplifiés ne permet pas le déploiement de solutions logicielles complexes à partir de ces langages. (faires des librairies par exemple)

LANGAGE CENTRÉ VARIABLES D'ARTICULATION

- Langages associés aux variables d'une animation, permettant de programmer leur variation dans le temps plutôt que de les définir uniquement graphiquement.
- Habituellement, on l'utilise comme langage d'appoint en supplément à un interface graphique fonctionnant avec des courbes et des trames d'animation. (Tel que vu au début du chapitre).
- Exemple: Flash et ActionScript





Adobe Flash CS2

LANGAGES GRAPHIQUES

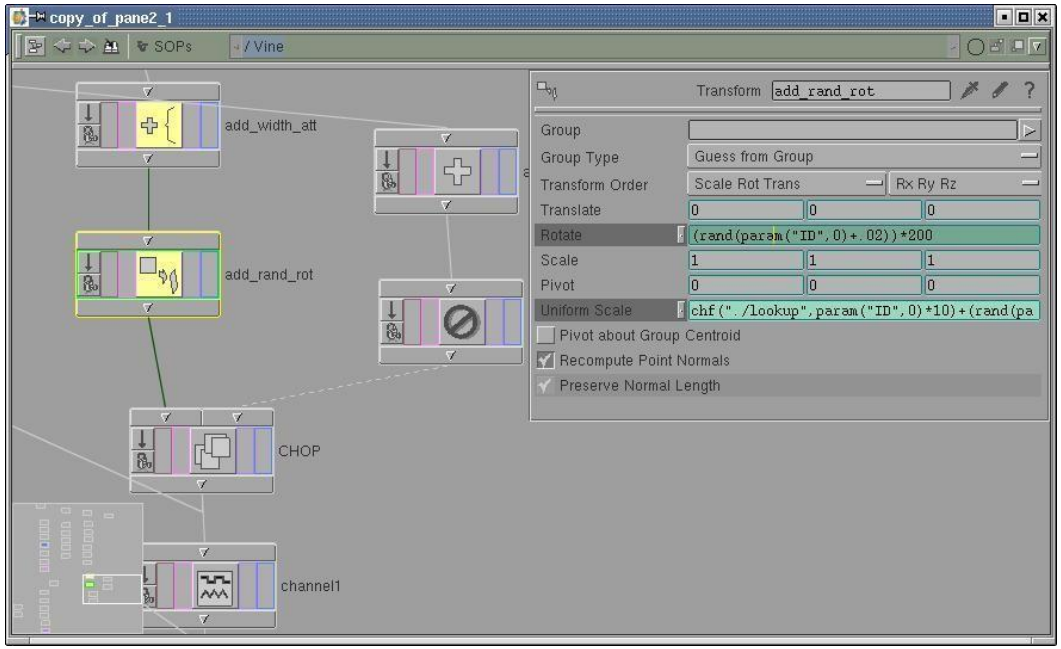
- Les langages graphiques utilisent une représentation graphique du flot d'une animation.
- Représenté sous la forme de graphe acyclique.
- Chaque noeud du graphe possède un nombre fixe de sorties et d'entrées et sert à faire une opération en particulier.
- Un noeud en soit possède des paramètres pouvant affecter l'effet qu'il aura sur l'animation.

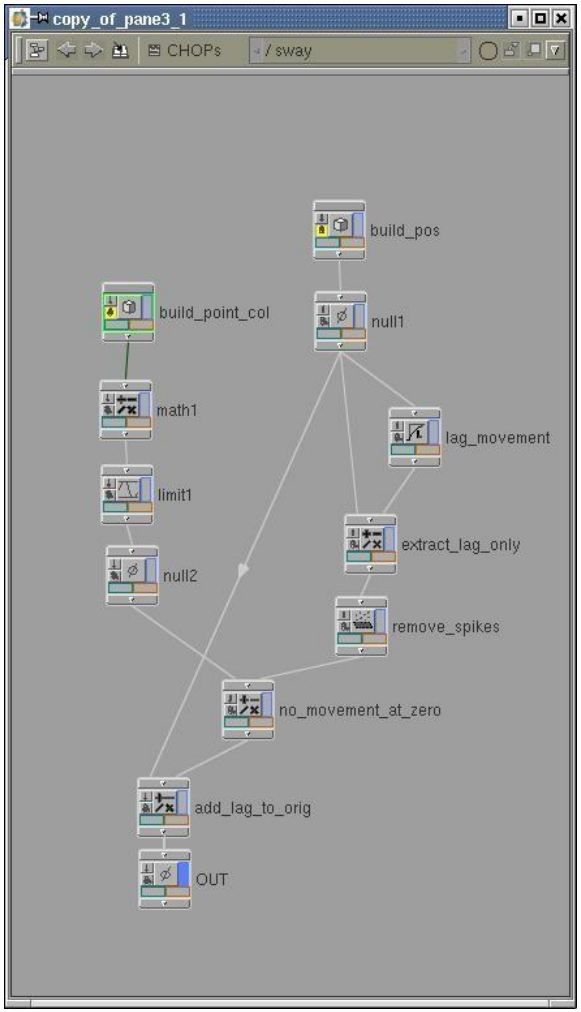
LANGAGES GRAPHIQUES

- Exemples de noeuds:
  - Noeud de transformation (servant à affecter l'orientation, la position, la couleur, etc.)
  - Noeud de modulation (servant à affecter la vitesse à laquelle est effectuée une transformation)
- On associe les noeuds de sortie à différentes parties du modèle 3D qui sont modifiées en fonction du graphe.

LANGAGES GRAPHIQUES

- Exemple de langage graphique:
  - Éditeur d'animation par graphe de *Houdini (Side effects software)*





Houdini, side effects software, 2008

LANGAGES PAR ACTEURS

- Équivalent des langages de programmation orientés-objets mais appliqués à l'animation.
  - Un acteur est l'équivalent d'un object.
    - Il contient les données relatives à ses propres animations (attributs) et les informations pour produire l'animation (méthodes).
    - Un acteur peut contenir d'autres acteurs
  - Exemple:
    - Un acteur "voiture" contient les acteurs "portes", "roue" et les animations requises pour monter/descendre les bancs, tourner les roues, etc.

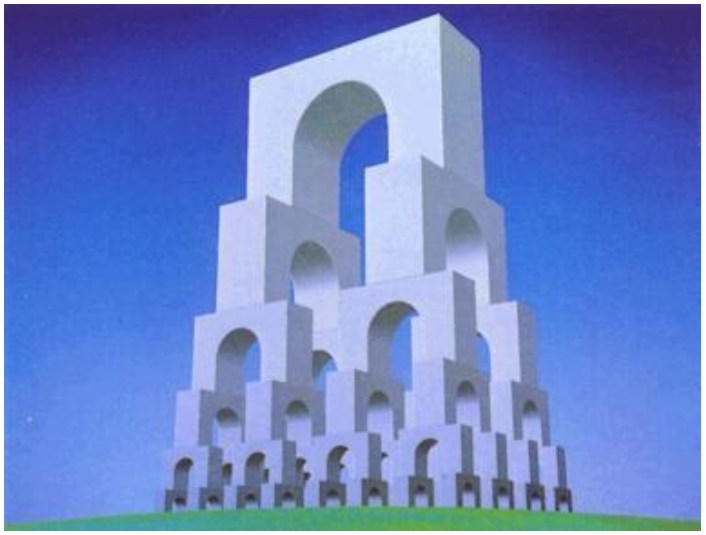
LANGAGES PAR ACTEURS

- Avantageux car il permet la réutilisation rapide de systèmes d'animation complexes.
- Permet aussi la complexification de systèmes déjà existants en rajoutant de nouvelles animations à un acteur déjà créé.
- On peut ainsi générer procéduralement des modèles complexes. (Par exemple avec des fractals)

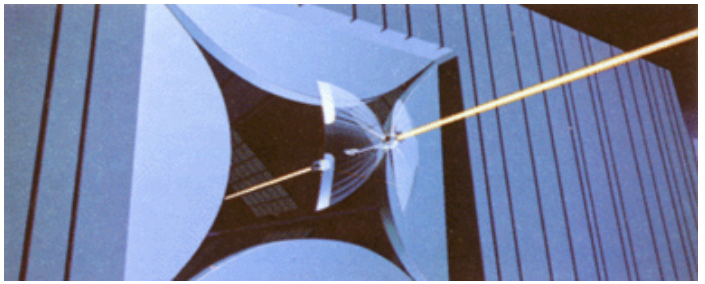
LANGAGES PAR ACTEURS

- Exemple :
  ASAS (Actor/script animation system)
  (C.W. Reynolds, 1982)

  - Très populaire durant les années 80.
  - Utilisé notamment pour construire la totalité des animations du film TRON.
  - Originalement basé sur LISP.



Reynolds, 1982



Tron, Disney, 1982

LANGAGES PAR ACTEURS

- Fonctionnalités du langage ASAS
  - Opérations géométriques (transformations)
  - Animation
  - Outils mathématiques d'animation
  - Passation de messages (entre les acteurs parents/enfants)

```
(script spinning-cubes
        (local: (runtime 96)
                (midpoint (half runtime)))
        (animate (cue (at 0)
                      (start (spin-cube-actor green)))
                 (cue (at midpoint)
                      (start (spin-cube-actor blue)))
                 (cue (at runtime)
                      (cut))))
```

Exemple de code ASAS pour animer un acteur.

```
(defop add-arch-level
       (param: sub-tower)
       (grasp sub-tower
              (scale fractal-ratio)
              (move (vector 0 height 0))
              (rotate 0.25 y-axis))
       (grasp arch-element
              (recolor (interp (quo levels total-levels)
                               bot-color
                               top-color)))
       (subworld (group arch-element
                        (move subtower-offset-1 sub-tower)
                        (move subtower-offset-2 sub-tower))))
```

Exemple de code ASAS pour la construction d'un acteur fractal

LANGAGES D'ANIMATION

- Avantages:
  - Formalise et fixe à des valeurs précises une animation. Elle peut donc être réécrite et rejouée de façon parfaitement identique.
  - Permet une approche algorithmique ou procédurale à l'animation.
- Désavantage:
  - Demande une connaissance et une compréhension de la programmation de la part de l'artiste. (L'artiste doit aussi être programmeur.)

SOMMAIRE, CHAP 4

- Animation par interpolation structurée
  - Cadre-clés classiques
  - Cadre-clés standards
  - Animation hiérarchique
    - Modélisation hiérarchique de scènes animées
    - Squelettes d'animation (articulations, arcs, transformations)
  - Skinning
    - Assignation directe
    - Linear blend skinning
    - Spherical blend skinning
      - Détermination du centre de rotation
  - Outils utilisés pour le skinning
- Langages d'animation (Annexe)
  - Langages de programmation complets
  - Langages orientés artiste
  - Langages centrés variables d'articulations
  - Langages graphiques
  - Langages par acteurs

RÉFÉRENCES

- L. Kavan, J. Zara, *"Spherical blend skinning: a real-time deformation of articulated models."*, Proceedings of the 2005 symposium on interactive 3D graphics and games, pp.9-16 (2005, Washington, D.o.Columbia)

- Reynolds, C. W. "*Computer Animation with Scripts and Actors",* in Computer Graphics, 16(3) (SIGGRAPH 82 Conference Proceedings) pp 289-296

- Parent,Rick : "*Computer Animation, Algorithms and techniques*", 3nd edition, Morgan Kaufmann, 2012