

L'animation en temps réel

... et révision des concepts

Animation et Multimédia

Sommaire

Les différentes méthodes de création d'une animation

- Procédurale
- Dynamique
- Keyframes

Les contraintes d'animation temps réel

Les LODs et les LOAs

Exemples

Animation et Multimédia

Introduction

L'animation par ordinateur dans les jeux vidéo date des 1^{ers} ordinateurs.

Elle joue un rôle prépondérant pour immerger l'utilisateur dans l'univers proposé

Tout d'abord réalisé en 2D par diverses techniques (scrolling, sprites, ...), elle a acquis aujourd'hui un réalisme qu'il n'était pas possible d'imaginer il y a 20 ans !



Animation et Multimédia

Animation procédurale

Animation définie par rapport à une action

Par exemple ajouter un comportement sur un objet qui le fait tourner sur lui-même

Permet de caractériser les propriétés d'un objet dans un jeu et son étroitement liées aux caractéristiques du jeu (réaction physique ou non, animation spécifique, ...)

Animation et Multimédia

Dynamique

Reproduire les phénomènes des lois physiques
Les objets sont animés uniquement par des lois physiques

Exemple : un personnage tenant une épée la lâche. La gravité s'applique sur cette épée fait qu'elle tombe au sol puisque aucune contrainte ne la retient.

Les jeux vidéo utilisent des lois similaires :

On a pas forcément besoin d'un comportement réel, mais simplement de faire croire qu'il est réel.

Équation fondamentale de la dynamique

$$F_1 + F_2 + \dots + F_n = m \cdot A$$

somme des forces

masse

accélération

Animation et Multimédia

Exemple de la gravité

On considère un objet où seul son centre de gravité reçoit l'application des forces. Ceci est suffisant pour simuler des lancers d'objets par exemple (sans tenir compte des forces de frottements). Dans ce cas on néglige toute rotation, seul un point (donc sans orientation) est considéré comme représentant l'objet.

Cette méthode est utilisée pour gérer les particules par exemple (représentant la poussière, le feu, ...)

La force appliquée :

$$P = m \cdot A$$

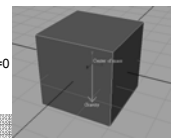
Nous permet de calculer l'accélération (constante) de l'objet par rapport au temps $\rightarrow A = P / m = g$

Ce qui nous intéresse, pour définir le mouvement, est la position de l'objet à un instant t . Pour la calculer, on intègre par rapport au temps l'équation de l'accélération (\rightarrow vitesse), puis on intègre la vitesse pour obtenir la position.

$$A(t) = g$$

$$V(t) = g \cdot t + V_0 \text{ où } V_0 \text{ est la vitesse au temps } t = 0$$

$$P(t) = \frac{1}{2} \cdot g \cdot t^2 + V_0 \cdot t + P_0, \text{ où } P_0 \text{ est la position initiale au temps } t=0$$



Animation et Multimédia

Ces formules peuvent s'appliquer pour des vecteurs 3D.

D'où

$A(t) = (0, -g, 0)$ la force étant verticale

$V(t) = (V_x(t), V_y(t), V_z(t))$ et $V_0 = (V_{0x}, V_{0y}, V_{0z})$

$P(t) = (P_x(t), P_y(t), P_z(t))$ et $P_0 = (P_{0x}, P_{0y}, P_{0z})$

Résolution théorique d'une équation différentielle linéaire du second ordre par rapport au temps t , avec second membre constant

$P(t)'' = (0, -g, 0)$

Comment résoudre ces équations ?

Animation et Multimedia

Approche itérative pour la résolution numérique

Consiste à calculer ces équations sur des intervalles de temps petits et en calculant une solution approchée sur chaque extrémité de l'intervalle.

Chaque nouvelle position 3D calculée avec sa vitesse permet de calculer la position et la vitesse approchée sur l'intervalle suivant

Utilisation des méthodes d'Euler et de Runge-Kunta

Exemple la méthode d'Euler (simple et rapide, mais approximative)

Cette méthode permet à un temps donné, connaissant la position et la dérivée (dans notre cas c'est vitesse) en ce temps de calculer la position et vitesse suivante. La formule est la suivante :

$NewPos = CurPos + h * CurSpeed$

Où h est la longueur de l'intervalle de temps utilisé.

Animation et Multimedia

On démarre à $t_0 = 0$:

$V(t_0) = V_0$ connu.

$P(t_0) = P_0$ connu.

On va considérer par exemple des intervalles de $1/30$ secondes (ce qui représente une frame sur deux pour un jeu tournant à 60 FPS).

Donc au temps suivant $t_1 = 1/30$, on aura

$V(t_1) = V(t_0) + (1/30) * A(t_0)$ avec $A(t_0) = A(t) = g$ constant par rapport à t .

$P(t_1) = P(t_0) + (1/30) * V(t_0)$

Puis au temps $t_2 = 2/30$, on aura :

$V(t_2) = V(t_1) + (1/30) * A(t_1)$

$P(t_2) = P(t_1) + (1/30) * V(t_1)$

Etc...

Animation et Multimedia

Keyframes

C'est le moyen le plus commun de création d'une animation.

Basée sur des « clés d'animation » (Keyframe)

Position (spatiale)

Rotation

Forme de l'objet

A partir des clés, on interpole les valeurs intermédiaires en les différentes formes, position ...

Dans le cadre d'un jeu vidéo (temps réel), le moteur graphique est capable de récupérer ces différentes informations de la phase de design, et de calculer les positions/formes intermédiaires

Différents types d'interpolations : linéaire, 2^{ème} ordre, ...

Animation et Multimedia

Algorithme général d'animation par keyframe

A partir d'un tableau contenant toutes les clés d'animation. A chaque clés est associé un temps (le temps où doit s'exécuter le changement de position)

Algorithme:

On prend en entrée le temps courant, t

On cherche entre quelles clés on se trouve dans le tableau → on obtient la clés inférieure Key_n et supérieure Key_{n+1} à des temps t_n et t_{n+1} ($t_n \leq t \leq t_{n+1}$)

A partir de ces 2 clefs, une fonction d'interpolation renvoie pour le temps courant t la valeur interpolée.

Remarque : la fonction d'interpolation peut utiliser d'autres informations que ces 2 clés, i.e. elle peut utiliser d'autres clés pour lisser l'animation

Animation et Multimedia

Interpolation linéaire

C'est le cas le plus simple.

On traite les clés 2 par 2, en considérant que nous avons une clé de départ et une clé d'arrivée qui sont des translations.

La valeur interpolée se trouve sur une droite entre les 2 positions

Soit P_n la position de départ à t_n , P_{n+1} la position d'arrivée à t_{n+1} .

Le segment de droite noté S entre nos 2 points est donné par sa formule paramétrique $S(t)$ qui est la suivante :

$S(t) = P_n + t * (P_{n+1} - P_n)$ avec t dans $[0, 1]$

Animation et Multimedia

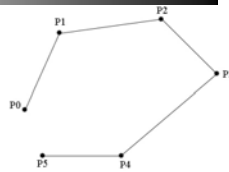
Exemple

Trajectoire d'une caméra

Rq: la caméra aura une vitesse de déplacement sur chaque segment pouvant être différente. La transition entre les différentes vitesses sera faite brusquement !

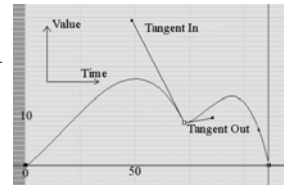
Car ... utilisation d'une interpolation du 1^{ère} ordre (ne prenant en compte que la position)

Pour remédier à cela, on introduit la notion de tangente à l'intérieur d'une clé d'animation. On rajoute donc 1 ou 2 tangentes à chaque clé. Ces tangentes vont donner des informations complémentaires sur la manière dont on arrive ou dont on part d'une clé. Ces informations seront la vitesse et la "direction".



Animation et Multimedia

Cette courbe d'animation comporte 3 clés qui sont représentées par les carrés noirs aux extrémités et le blanc au milieu. La courbe joignant les 3 clés est ce que l'on appelle la courbe ou trajectoire d'animation. Sur la 2^{ème} clé, on peut visualiser, 2 tangentes In et Out.



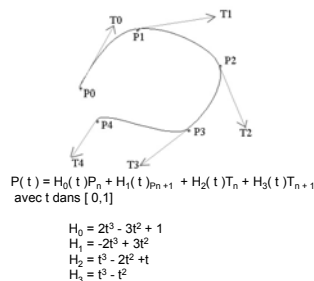
Dans certains cas, plutôt que d'éditer directement les tangentes sur la courbe, il est possible de rentrer certains paramètres qui seront utilisés pour reconstruire les tangentes avec l'aide des clés voisines.

Animation et Multimedia

Exemple : courbe d'Hermite

L'interpolation de type Hermite est une interpolation qui fait intervenir des valeurs de **clés**, et des **tangentes**. Dans le cas le plus général d'Hermite, on ne prend en compte qu'une tangente par clé.

Sur cette courbe d'Hermite, on a 5 points notés de P0 à P4, en chacun de ces points part une tangente avec en son extrémité une flèche, les tangentes sont notées de T0 à T4. On voit que la courbe résultante passe par tous les points.



Animation et Multimedia

Concept ...

Valable pour tous les types d'animation / mouvement :

Position → Vitesse → Accélération

Animation et Multimedia

Les contraintes de la modélisation pour le temps réel

Pour une modélisation parfaite il faut un mesh le plus proche possible du design

Une solution :

Utiliser des facettes, des surfaces de subdivision, des NURBS/patch ou des meshes les + lisses possibles
Donc contenant des centaines de milliers de faces !

Cela pose des problèmes

De capacité graphique
De taille de données à traiter
De manipulation de données

A effectuer en temps réel ! ☹

Animation et Multimedia

Un mesh pour le temps réel impose des contraintes, dépendantes de plusieurs critères comme

Les capacités graphiques de la plate forme cible
Le moteur de rendu utilisé
Le type de jeux ou d'application
Transmission des données ou pas (jeux en réseau, application distante, ...)

Il n'existe pas de critères spécifiques !

solution(s): combinaisons de techniques et de contraintes

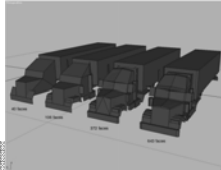
Animation et Multimedia

Lorsque vous modélisez un mesh temps réel, vous devez le faire en fonction des contraintes de la plateforme cible. Ces contraintes sont très différentes si vous ciblez une PS2 et un Xbox par exemple. Vous devez donc connaître les spécificités techniques de votre cible (capacité mémoire, processeur graphique ...)

De plus, vos meshes vont être affichés par un moteur de jeux vidéo, là aussi chaque moteur possède ses spécificités et souvent chaque moteur préfère un mesh sous une certaine forme plutôt qu'une autre... Généralement, un mesh formé par un ou plusieurs strips de triangles.

Des techniques permettent de « contourner » certains problèmes posés par la réduction des polygones (Bump mapping)

Rq : certains calculs sont maintenant pris en compte directement dans les processeurs graphiques (GPU)



Animation et Multimedia

Contraintes sur les meshes et la scène / LOD / Subdivisions spatiales / Mémoire

Toute scène affichée à l'écran devra avoir un nombre d'image par secondes (FPS pour Frame Per Second) raisonnable, généralement supérieur à 15 images secondes, idéalement à 60 images /secondes pour les jeux, 25 pour des animations classiques.

Pour éviter ce genre de problèmes, il nous faut limiter le nombre de meshes d'une scène 3D. Précisément, limiter le nombre de triangles par frames envoyés à l'affichage. Par exemple sur PC/XBox, l'affichage est fait par votre carte 3D. Sur PS2, par exemple, c'est un mélange entre différentes composants dont le GS (Graphic Synthesiser).

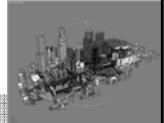
⇒ définition d'un budget de faces / vertices doit être défini **avant** la modélisation.

Il y a une autre technique utilisée couramment dans la modélisation temps réel qui est le LOD (level of detail). Il s'agit d'avoir plusieurs versions d'un même mesh qui sont des versions simplifiées de ce mesh.

Pourquoi faire cela ?

Un objet composé d'un grand nombre de polygones peut représenter que qqs pixels sur l'écran ... autant simplifier les calculs !!

Pour générer des LOD, il existe 2 principales techniques : les LODs statiques et dynamiques.



Animation et Multimedia

Les LODs statiques

Les LODs statiques consistent à définir/créer plusieurs meshes avec un niveau de détail différent

Il y a le mesh original, qui est le plus complexe en nombre de faces et différents niveaux de simplifications de ce mesh qui peuvent aller de 1 à n, suivant la mémoire disponible pour stocker vos meshes et leurs LODs.

Ensuite durant la phase d'affichage, suivant la distance du mesh à la caméra courante, le moteur graphique utilise telle ou telle *version* du mesh.

Remarque

On utilise souvent de l'interpolation (du morphing) entre 2 LODs afin de ne pas créer un effets d'apparition/disparition trop brutal lors du changement de niveau de détail d'un mesh, et pour réduire la charge de travail au niveau du modeling.

Le LOD statique nécessite peu de calculs mais il faut créer (souvent manuellement) les différentes versions simplifiées du mesh. Et au chargement de la scène, il faut stocker en mémoire toutes les versions simplifiées des mesh ou les streamer.

Animation et Multimedia

Les LODs dynamiques

Il existe aussi des algorithmes utilisables en temps réel qui en partant d'un mesh original peuvent le simplifier automatiquement en fonction de sa distance avec la caméra courante.

⇒ « **Continuous LOD** ».

Ils sont plus gourmand en temps de calcul machine, car ces opérations demandent des calculs complexes, mais qui ne nécessitent pas de stocker les différentes versions simplifiées du mesh.



Animation et Multimedia

Il existe aussi des « optimisations » au niveau de l'animation et non pas seulement du modeling (voir ci dessous)

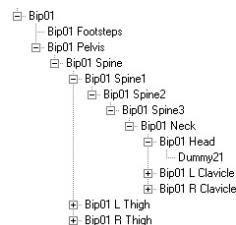
Exemple :

l'animation faciale

- réduction du nombre de paramètres à tenir en compte

le squelette

- regroupement de sous segments



Animation et Multimedia

Animation d'un squelette

Notion de pose : Une pose c'est l'ensemble des matrices de transformation des bones à un instant donné. Elles permettent de définir l'état du squelette de manière unique.

On parle souvent de pose initiale, c'est celle dont on s'est servi pour associer le mesh au squelette. Dans cette pose spéciale, le mesh n'est pas déformé par le squelette.

La cinématique directe et inverse : on parle de cinématique directe lorsque l'on répercute le mouvement d'un bone sur tous ses fils. Dans le cas où l'on bouge un bone et que l'on répercute le mouvement sur tous ses parents, on parle de cinématique inverse. On l'utilise par exemple lorsque une main se trouve sur un tiroir, au lieu de faire en sorte que la main tire le tiroir, c'est le tiroir qui pousse la main et on utilise de la cinématique inverse...

La répercussions des mouvements du squelette sur les vertices du mesh est appelée *skinning*

Animation et Multimedia

Skinning

On appelle ce processus skinning car il est utilisé pour déformer un mesh représentant la peau du personnage par l'animation d'un squelette.

Comment fait-on le lien entre le squelette et le mesh ? Il faut que si le bone représentant un bras bouge, les vertices du mesh « proches » du bras bougent en même temps pour représenter sur le mesh la déformation.

Pour cela, on différencie 2 types de skinning le **rigid** et le **smooth** (appelé aussi blended).

Animation et Multimedia

Rigid skinning

Définition

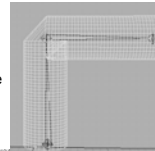
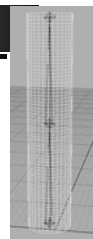
On parle de **rigid skinning** lorsque chaque vertex du mesh n'est influencé que par un et un seul bone à la fois.

Pseudo-algorithme (non optimisé). Lorsque un bone est animé :

- on récupère le delta de rotation du bone.
- on transforme ce delta dans le repère du mesh.
- on récupère les vertices influencés par ce bone dans le mesh.
- on applique le delta sur chaque vertex et sur chaque normale au vertex.

Ceci nécessite d'avoir pré calculé par quel bone est influencé chaque vertex du mesh. En général, cette opération est faite en pré processing, on récupère directement la correspondance vertex → bone dans un moteur graphique.

Le **rigid skinning** est peu coûteux mais il a un défaut visible sur les figures suivantes.



Animation et Multimedia

Smooth skinning

On utilise non plus un seul bone pour influencer un vertex mais plusieurs.

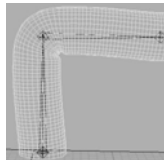
Chacun de ces bones a une influence appelée "un poids" sur un vertex. La somme des poids des bones déformant un vertex fait toujours 1.

Exemple : si on veut qu'un vertex noté V soit influencé à 70% par un bone A et à 30% par un bone B, ces bones auront respectivement un poids de 0,7 et 0,3 par rapport à V.

Remarque : Un poids de 1 indiquerait que seul ce bone influencerait le vertex (rigid skinning). Un poids de 0 indique que le bone n'influencerait pas du tout le vertex.

Dans le **rigid skinning**, nous avons une correspondance 1 vertex → 1 bone.

Dans le **smooth skinning**, nous avons 1 vertex → n bones avec n poids W_i tels que $W_0 + W_1 + \dots + W_n = 1$.



Animation et Multimedia