Introduction à la synthèse d'animation par ordinateur

François Faure

6 février 2001

1 Généralités

Animation: simulation du mouvement par la succession rapide d'images fixes.

Fréquences utilisées:

- premiers films: 16 Hz
- actuellement: 25 ou 30 Hz
- télévision (images entrelacées): 50 Hz (PAL/SECAM), 60 Hz (NTSC)
- quasi-perfection à 70 Hz

Outils de production

- caméra
- dessin manuel
- images de synthèse

Principaux avantages de l'ordinateur:

- production de masse (30 Hz, 90 min $\rightarrow$ 162000 images)
- scènes virtuelles, choix du rendu
- interactivité (jeux, simulateurs)
- gestion du scénario

Applications:

- films
  - divertissement
  - publicité
  - éducation
- jeux:
  - interactivité (ping-pong)
  - créatures animées (pac man)
  - mondes complexes (simcity, quake)

    - joueurs en réseau (quake)
- simulateurs:
    - apprentissage
        - conduite (avions, voitures)
        - manipulation (telemanipulation, chirurgie)
    - conception assistée par ordinateur (CAO)
        - robotique
        - ergonomie
- visualisation
    - phénomènes physiques (déformation, écoulements)
    - mondes virtuels
        - non immersifs (VRML)
        - immersifs (réalité virtuelle, réalité augmentée)

2 Contrôle du mouvement

Une scène se compose d'éléments constants (forme d'un solide, couleur,...) et d'éléments variables (position d'un solide, points de contrôle d'une surface déformable, couleur,...) qui nous intéressent en animation et que nous appelons **paramètres de contrôle**. Les valeurs des paramètres de contrôle définissent entièrement l'**état du système** à un instant donné.

Les paramètres de contrôle sont gérés par des **contrôleurs**, modules logiciels qui gérent les valeurs des paramètres au cours du temps. Suivant les cas, leur complexité structurelle varie de la simple fonction au réseau d'automates hiérarchiques.

Soit q l'ensemble des paramètres de contrôle: valeurs dont l'évolution définit le mouvement de l'image à afficher. Appelons t le temps et e les entrées du système en provenance par exemple de l'utilisateur. On distingue différentes classes de contrôle du mouvement:

- par guidage: l'utilisateur pilote directement les mouvements (souris, capture de mouvement). On a $q = f(e)$
- par mouvements prédéterminés: le programme a les données nécessaires pour afficher la scène à n'importe quelle date. On a $q = g(t)$
- par modéles générateurs: à tout instant on sait calculer, plus ou moins précisément, la variation de l'état. On $\delta q = h(q,e,t)$

Ces classes ne sont pas étanches et une même application peut combiner différents types de contrôle selon les besoins.

3 Boucle d'animation

3.1 Principe

La boucle d'animation permet de continument remettre à jour les paramètres et réafficher, comme illustré sur la figure 1.

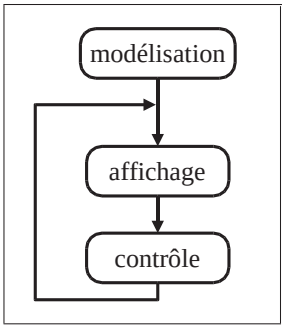

FIG. 1 – *Boucle d'animation*

En général, le contrôle consiste à lire les éventuelles entrées du système, puis à mettre à jour les paramètres en suivant des règles prédéfinies. Toutefois une part de (re)modélisation peut intervenir dans certaines applications.

3.2 Implémentation avec GLUT

GLUT, comme beaucoup d'autres APIs, permet au programmeur de définir sa boucle d'animation sous forme de *callbacks*: méthodes dont l'exécution est déclenchée dans certaines conditions. Le programmeur ne maitrise pas les conditions d'exécution mais seulement la nature de celle-ci. GLUT transmet les paramètres nécessaires. La table 1 présente les principaux évènements détectables par GLUT, la méthode glut pour définir le callback adéquat (par l'intermédiaire d'un pointeur de fonction), ainsi que les paramètres transmis.

évènement à traiter	**fonction correspondante**	**paramètres transmis**
redimensionnement fenêtre	glutReshapeFunc()	int w, int h
affichage requis	glutDisplayFunc()	-
click de souris	glutMouseFunc()	int b, int state, int x, int y
mouvement de souris	glutMotionFunc()	int x, int y
frappe du clavier	glutKeyboardFunf	int key, int x, int y
frappe touche spéciale	glutSpecialFunc	int key, int x, int y
inaction	glutIdleFunc()	-

TAB. 1 – *Principaux callbacks de GLUT.*

La fonction spécifiée dans glutIdleFunc() permet d'appliquer les contrôleurs. La méthode glutPostRedisplay() émet un signal provoquant le réaffichage.

L'extrait de programme suivant déplace un point dans l'espace, à vitesse et direction constantes. Le mouvement s'interrompt quand on appuie sur le bouton gauche de la souris, et reprend quand on relache.

```
double pos=0, increment=1;

void display(){
  ...
```

```
  glBegin(GL_POINTS)
  glVertex3f(pos,0,-10);
  glEnd();
}

void update() {
  pos +=increment;
  glutPostRedisplay();
}

void mouse(int button, int state, int , int ){
  if( button == GLUT_LEFT_BUTTON && state == GLUT_DOWN )
    increment = 0;
  else if( button == GLUT_LEFT_BUTTON && state == GLUT_UP )
    increment = 1;
}

main(){
  ...
  glutDisplayFunc(display);
  glutMouseFunc(mouse);
  glutIdelFunc(update);
  glutMainLoop();
}
```

3.3 Notion de temps

La notion de temps varie grandement suivant le type d'animation. En animation guidée, aucune représentation du temps n'est nécessaire. Sinon, on utilise une variable qu'on met à jour à chaque passage dans la boucle d'animation (*temps simulé*). Les applications dites **temps-réel** garantissent que le temps simulé s'écoule à la même vitesse que le "vrai" temps, et qu'un nombre bien défini de passages dans la boucle (typiquement entre 12 et 60) sera effectué dans chaque seconde. Si l'intervalle entre chaque image est supérieur au temps de calcul nécessaire, le programme attend, comme illustré sur la figure 2.

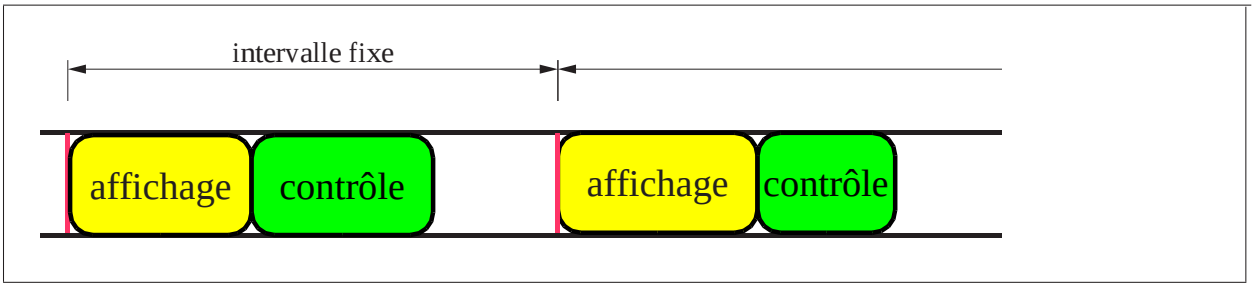

FIG. 2 – *Exécution d'une application temps réel.*