

Créer un moteur de déplacement 2D d'une voiture sous Flash MX

par [Julien DEFAUT](#)

Date de publication : 10/03/2005

Dernière mise à jour : 10/03/2005

Dans ce tutorial, nous allons réaliser un moteur de déplacement 2D d'une voiture dirigée par les touches directionnelles. Le tout sera fait sous Flash MX donc en Actionscript 1. Téléchargez la version pdf.

Preview

- 1 - Créations des clips
- 2 - Initialisations
- 3 - Gestion de la rotation
- 4 - le déplacement
- 5 - Les collisions

Preview

Voici le résultat attendu de ce tutorial (utilisez les touches fléchées) :

<http://download.macromedia.com/pub/shockwave/cabs/flash/swflash.cab#version=6,0,0,0>
400 300 movie

1 - Créations des clips

- Créez une scène vide et placez-y 2 Clips. L'un représente la voiture et l'autre l'obstacle. Pour créer un Clip, vous dessinez une forme que vous sélectionnez puis F8.
- Dans la fenêtre qui apparaît, vous indiquez un nom en cochant la case "Clip" puis "OK".
- Cliquez sur l'obstacle et dans la fenêtre "Propriété", indiquez juste en dessous de la liste déroulante avec le mot "Clip", le nom de variable suivant : "rond". Il nous permettra d'intervenir sur le clip obstacle en ActionScript de n'importe où dans l'application.

2 - Initialisations

Cliquez sur la voiture, faites F9, passez en mode Expert (dans le petit icône en haut à droite de la fenêtre Actionsript sur la barre grise), ce sera plus pratique pour rédiger du code.

Placez dans premier temps, le code suivant dans la fenêtre ActionScript :

```
onClipEvent(load)
{
    vitesse=0;
    x=0;
    y=0;
}
```

Ceci signifie que lors du tout premier chargement du clip (la voiture), nous initialisons 3 variables à 0. Ce n'est pas obligatoire, car par défaut elles sont déjà à 0 mais sous Flash 2004 et dans beaucoup d'autres langages de programmation, ce n'est pas le cas. Pour ces raisons et d'autres, mieux vaut les initialiser manuellement.

3 - Gestion de la rotation

Sous le `onClipEvent(load) { }`, placez :

```
onClipEvent (enterFrame) // le code suivant est exécuté en boucle (il s'agit d'une sorte de timer en flash)
{
    if (Key.isDown(Key.UP)) vitesse += 1; //accélération en avant
    if (Key.isDown(Key.DOWN)) vitesse -= 0.2; //accélération en arrière
    if (Key.isDown(Key.LEFT)) _rotation -= 8; //rotation affecte directement la voiture
    if (Key.isDown(Key.RIGHT)) _rotation += 8;

    if (Math.abs(vitesse)>1) vitesse *= 0.9; // limite l'accélération quand la vitesse augmente trop
}
```

Ce code sera exécuté à chaque passage du curseur de la timeline sur la frame contenant le Clip, il n'y a qu'une frame, donc par défaut, ce sera 12 fois par seconde. Le mieux pour la fluidité est d'aller dans `Modification>Document` et de passer à 30 images/seconde par exemple.

Les commentaires contenus dans le code semblent assez explicites. Mais dans l'état actuel du code, seule la rotation fonctionnera puisque `_rotation` est une variable propriétaire aux objets Clip, tandis que `vitesse` est une valeur juste définie par nous.

Le déplacement visuel va donc se faire dans un second temps.



Remarque : en ActionScript (et comme tous les langages avec une syntaxe proche du C), la notation `variable+=3` est équivalente à `variable=variable+3`.

4 - le déplacement

Mettez à jour le code précédent en ajoutant 4 lignes de code :

```
onClipEvent (enterFrame) // le code suivant est exécuté en boucle (il s'agit d'une sorte de timer en flash)
{
    if (Key.isDown(Key.UP)) vitesse += 1; //accélération en avant
    if (Key.isDown(Key.DOWN)) vitesse -= 0.2; //accélération en arrière
    if (Key.isDown(Key.LEFT)) _rotation -= 8; // _rotation affecte directement le carré bleu
    if (Key.isDown(Key.RIGHT)) _rotation += 8;

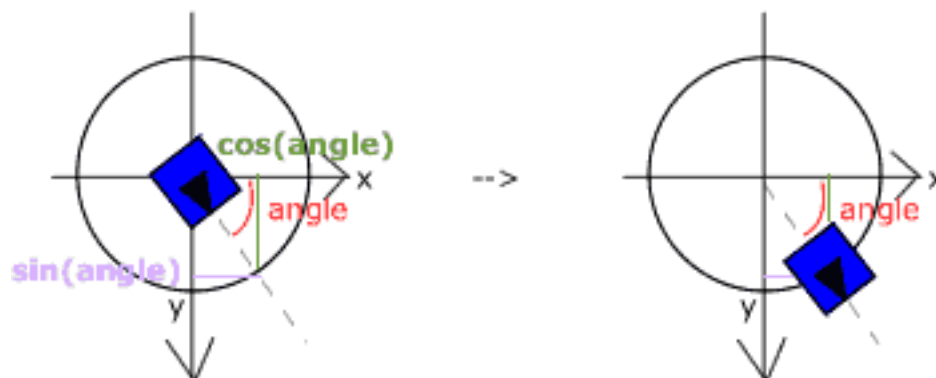
    if (Math.abs(vitesse)>1) vitesse *= 0.9; // limite l'accélération

    x = Math.cos(_rotation*(Math.PI/180))*vitesse; // donne la position x future en fonction de la vitesse
    y = Math.sin(_rotation*(Math.PI/180))*vitesse; // donne la position y future en fonction de la vitesse

    _x += x; // _x est une coordonnée du carré bleu
    _y += y; // _y est une coordonnée du carré bleu
}
```

Le centre de la voiture est positionné dans la scène selon une abscisse `_x` et une ordonnée `_y`. Vous pouvez donc, en changeant la valeur de `_x` ou `_y`, modifier la position de la voiture. Cependant, lorsque `_rotation` change, la voiture avancera dans la direction donnée par l'angle `_rotation`. Et quand on parle d'angle et de coordonnées, l'outil mathématique souvent indispensable est la trigonométrie.

A chaque chargement de notre bloc de code `onClipEvent (enterFrame){ }`, la voiture devra avoir une nouvelle position. Le but est de calculer la position future en fonction de la vitesse et de la `_rotation`.



$$\text{angle} = _rotation * \text{PI}/180$$

On raisonne dans le cercle trigonométrique (ci-dessus) qui possède un rayon de 1 et dont les angles s'expriment en radians (PI radians = 180 degrés).

Pour convertir `_rotation` qui s'exprime en degrés, on effectue donc la conversion suivante "`angle = _rotation * PI/180`".

La taille de l'avancée selon `x` future sera ici `cos(angle)` et pour `y` : `sin(angle)`. La position future est le

point de jonction de la ligne pointillée et du cercle sur le schéma.

Cependant \cos et \sin sont toujours inférieurs ou égaux à 1. Elle n'avancera pas vite notre voiture. Il vaut mieux donc multiplier les résultats par un coefficient qui augmentera le mouvement, c'est notre variable "vitesse" !

Nous avons donc :

```
x = Math.cos(_rotation*(Math.PI/180))*vitesse;  
y = Math.sin(_rotation*(Math.PI/180))*vitesse;
```

Comme nous sommes partis du principe que l'origine du repère est le centre de la voiture, il convient d'ajouter à la position précédente notre nouvelle valeur d'où :

```
_x+=x;  
_y+=y;
```

Si vous testez, la voiture se déplace ! Mais aucune collision n'est gérée ... Passons alors à l'étape suivante.

5 - Les collisions

Le code suivant sera, pour ce tutoriel, défini :

```
onClipEvent (enterFrame) // le code suivant est exécuté en boucle (il s'agit d'une sorte de timer en flash)
{
    if (Key.isDown(Key.UP)) vitesse += 1; //accélération en avant
    if (Key.isDown(Key.DOWN)) vitesse -= 0.2; //accélération en arrière
    if (Key.isDown(Key.LEFT)) _rotation -= 8; //rotation affecte directement le carré bleu
    if (Key.isDown(Key.RIGHT)) _rotation += 8;

    if (Math.abs(vitesse)>1) vitesse *= 0.9; // limite l'accélération

    x = Math.cos(_rotation*(Math.PI/180))*vitesse; // donne la position x future en fonction de la vitesse
    y = Math.sin(_rotation*(Math.PI/180))*vitesse; // donne la position y future en fonction de la vitesse
    if (_root.rond.hitTest(_x+x,_y+y,true)) // collision avec le rond
        vitesse *= -0.5; // réaction après collision, rebond de la voiture
    else
    {
        _x += x; //_x est une coordonnée du carré bleu
        _y += y; //_y est une coordonnée du carré bleu
    }
}
```

Nous avons créé un Clip obstacle pouvant être appelé par son nom : "rond".

Pour détecter une collision entre 2 Clips, Flash dispose d'une fonction TRÈS pratique : hitTest.

A chaque passage dans le bloc enterFrame, nous allons donc tester si la position future de la voiture créera une collision ou non : `_root.rond.hitTest(_x+x,_y+y,true)` renvoie true s'il y aura collision et false sinon.

Si il n'y a pas de collision on avance la voiture comme à l'étape 4. En cas de collision, on inverse légèrement le mouvement de la voiture pour simuler le rebond : `vitesse *=-0.5`;

La conséquence de cette affectation apparaîtra dans le prochain passage dans le bloc (il faut recalculer x et y en fonction de la nouvelle vitesse).

Tout fonctionne et, maintenant, vous pouvez étendre les fonctionnalités du jeu. [Téléchargez](#) la source en fla de ce tutoriel.