



Projet : Jeu pédagogique en Flash Etude Technique

Référence du document	Cahier charge.doc	version	1.2
Auteurs création	Sami Ben Hassine Fares Nabi	Date de création	24/04/2006 20:34:00
Auteur modification		Date de modification	

Introduction.....	2
Contraintes	2
Présentation.....	2
Légende.....	2
Commandes.....	3
Calcul de la note.....	3
Limiter la zone du défilement du viseur.....	3
Zoom et rotation.....	4
Zoom par zone.....	4
Conversion des coordonnées du local vers le global	4
Affichage du guide	5
Capture de la photo.....	5
Conclusion	6



Etude Technique

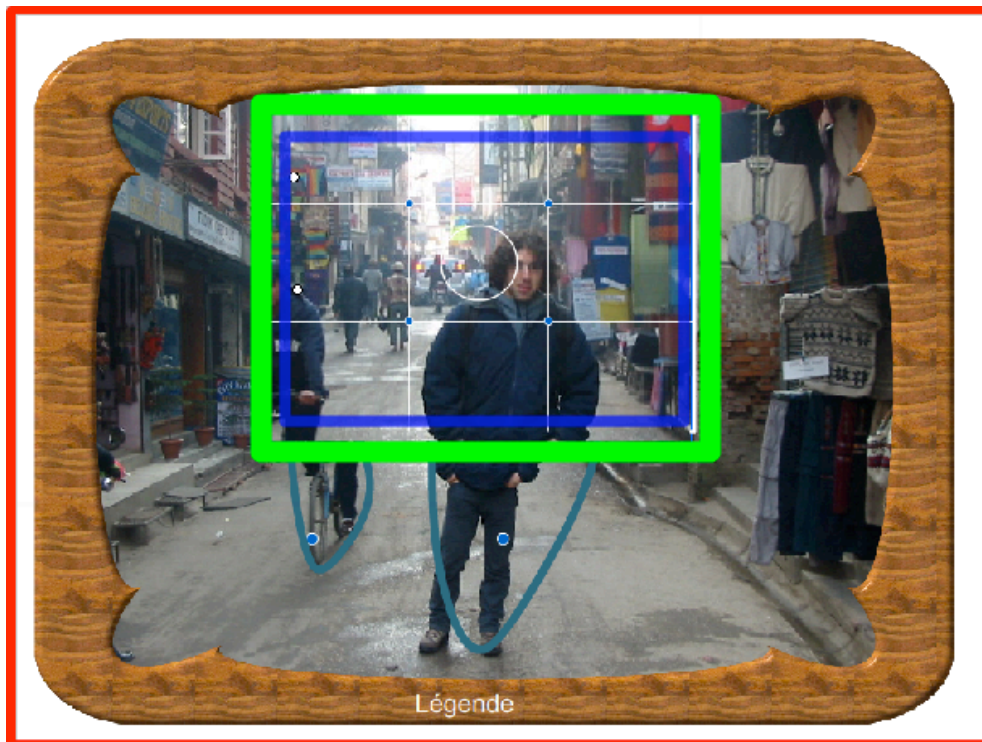
Introduction

Nous allons dans cette étude technique présenter les différents problèmes rencontrés lors de la conception du prototype et les différentes solutions qu'on a choisies en fonction des contraintes.

Contraintes

- Contrainte de temps :
Semestre de printemps pour l'ensemble du projet (du 15 mars 2006 au 2 juin)
- Contrainte de performance:
 - Résolution d'écran standard 800 x 600.
 - Le lecteur Flash Player 8 est nécessaire
 - La programmation se fera en ActionScript 2.0
- Contrainte de taille:
 - Taille du fichier SWF ne dépassant pas les 3 MO.

Présentation



Fenêtre principal du prototype

Légende

Rouge : fenêtre principale de l'application ou stage.

Vert : Movie Clip de la caméra.

Bleu : Movie Clip du viseur situé à l'intérieur du MovieClip de la caméra.

MC : MovieClip dans la suite de ce texte



Commandes

Flèches directionnelles : Zoom et rotation.
Clic gauche : prise de photo ou score.

Calcul de la note

Le calcul se fera sur trois axes :

- Un point représentera la tête du sujet à photographier, plus l'un des 4 points de rencontre en est proche plus la note est améliorée.
- Lorsque la ligne de tiers de haut est proche de la ligne d'horizon, la note est améliorée.
- Lorsque les deux lignes représentant les deux sujets sont proches des lignes de tiers verticales, la note est améliorée
- Lorsqu'il l'utilisateur effectue une rotation, si elle n'est pas verticale, il obtiendra la note E car l'objectif de la première mission n'est pas d'effectuer des rotations, mais de placer des objets sur des lignes de tiers.

Le calcul de la distance entre un point et l'un des 4 autres points utilise le théorème de Pythagore.

```
distancefromthis1 = Math.round(Math.sqrt((xdistp1*xdistp1)+(ydistp1*ydistp1)));
```

Le calcul de la distance entre les lignes est différent selon que l'image soit horizontale ou verticale. Le score est basé sur la ligne qui est la plus proche de notre ligne de tiers, on calcule la différence entre les coordonnées x ou y de la ligne de la photographie et de la ligne de tiers.

L'utilisateur pourrait obtenir un bon premier Score1 car il a par exemple mis le sujet sur une des lignes de tiers, mais un mauvais Score2 car il n'a pas placé la tête sur un des points forts.

La moyenne de ces trois scores, calculés séparément, est la note de l'utilisateur. Ce calcul est un pourcentage qui sera converti en lettres (A B C D E F).

Limiter la zone du défilement du viseur

Le curseur de la souris se transforme en viseur lorsqu'il entre dans la fenêtre du jeu, mais il fallait limiter la portée du curseur en limitant ses mouvements dans un espace fermé. On a opté pour StarDrag au début, mais cette solution se révéla rapidement limitée, car le viseur pouvait aller n'importe où sur le Stage.

Nous avons ensuite opté pour :

```
follow.onEnterFrame = function():Void {  
    this._x += (Math.max(bounds._x, Math.min(_xmouse, bounds._x+bounds._width-this._width))-this._x);  
    this._y += (Math.max(bounds._y, Math.min(_ymouse, bounds._y+bounds._height-this._height))-this._y);  
};
```

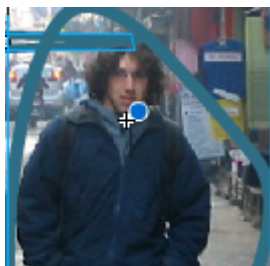
Bounds est un MC constituant les frontières à ne pas dépasser. Il s'agit d'un carré de la taille du Stage. follow est le MC qui remplacera la souris, ainsi on assigne le x_mouse et le y_mouse aux coordonnées x et y du MC follow. Mais uniquement dans le cas où ce x_mouse et y_mouse est inférieur aux coordonnées Bounds. Dans le cas contraire, le MC ne bouge plus.




Zoom et rotation

Le zoom ne s'effectuait pas comme prévu au début, au lieu de faire zoomer l'image du centre vers l'extérieur de l'écran, le zoom s'effectuait sur une partie et la photo sortait alors de l'écran. On a pensé à centre l'image après chaque rotation et après chaque zoom.

Puis on a trouvé une solution plus radicale, il suffisait de placer le registration point au centre de l'image. La rotation est aussi corrigée de cette manière, puisque l'image tourne autour de son centre de gravité.



Le registration point est la croix du milieu, pour modifier son emplacement, on appuie sur l'icône 

Zoom par zone

Le zoom implémenté étant un zoom sur toute l'image, on a pensé à l'améliorer en implémentant un zoom par partie parfaitement configurable. Un peu à la manière d'une loupe qui parcourt un texte.

La solution était de créer un masque dans le MC de la caméra. Puis dans le layer juste au-dessous du masque, de placer la même image à photographier, mais avec une taille plus grande.

Un nouveau problème s'est présenté alors présenté: on ne peut pas utiliser la méthode StarDrag avec un MC contenant un masque directement à partir du _root, il fallait donc implémenter cette méthode directement au MC du viseur contenu dans le MC de la caméra.

```
follow.glass.onEnterFrame= function() {  
    startDrag("follow");  
}
```

Conversion des coordonnées du local vers le global

Lorsqu'on a souhaité effectuer les calculs de note, les coordonnées qui nous été retournés étaient calculés par rapport aux MC et non par rapport aux coordonnées du stage, on a eu donc à utiliser la méthode localToGlobal pour convertir du local vers le globale.

Pour cela, on commence par créer un objet avec les coordonnées local qu'on désire convertir au global. Puis d'utiliser la méthode localToGlobal sur cet objet. Notez qu'il doit avoir des coordonnées nommés `x` et `y` non `_x` et `_y`.



```
var p1 = {x:follow.p1._x, y:follow.p1._y};  
var p2 = {x:photo1_mc.p1._x, y:photo1_mc.p1._y};  
follow.localToGlobal(p1);  
photo1_mc.localToGlobal(p2);  
xdist = Math.round(p1.x-p2.x);  
ydist = Math.round(p1.y-p2.y);  
distancefromthis = Math.round(Math.sqrt((xdist*xdist)+(ydist*ydist)));
```

Affichage du guide

Le problème posé était d'afficher un guide lorsque le cercle central touchait certains objets puis de le faire disparaître au bout d'un moment, on a utilisé la fonction *setInterval* qui se lance dès que la souris traverse l'objet, puis attends 500 ms avant de lancer la fonction *clearsujet* qui cache l'objet en le rendant transparent.

Au début, on a changé les propriétés *_visible* vers *false* pour cacher l'objet mais on s'est rendu compte que dans ce cas l'objet ne répond plus aux événements (*onRollOver*, *onMouseUp*).

```
75 photo1_mc.sujet1.onRollOver = function() {  
76     clearInterval(c1);  
77     photo1_mc.sujet1._alpha = 100;  
78     c1 = setInterval(clearsujet, 1000);  
79 };
```

Capture de la photo

L'implémentation d'un album qui affichera les images prises par l'utilisateur à la fin du jeu n'a pas été aisée. Nous avons au début pensé à la fonction *DuplicateMovieClip* utilisé en concert avec un masque. Il fallait alors stocker les coordonnées x, y à chaque fois. Mais cette solution s'avéra difficile à gérer quand il y a un stock important de photo à implémenter.

La version 8 de Flash incorporait une nouvelle bibliothèque *BitmapData* qui permettait justement d'effectuer des captures du Stage. On a donc créé un objet *BitmapData*, puis on a choisi une partie de l'image avec une matrice de translation, ensuite on a effectué la capture avec la méthode « draw », et enfin on l'a rattaché à un MC. Attacher l'objet *BitmapData* à un MC est indispensable, sinon l'image ne sera pas affichée.

Le problème avec la méthode *DRAW*, c'est que lorsqu'on effectue des modifications en temps réel du MC qu'on veut capturer, ces modifications ne seront pas répercutées sur la capture : la copie s'effectue toujours à partir de l'image originale.

Pour résoudre ce problème, nous nous sommes rendu compte que le deuxième paramètre de la méthode *DRAW* est une matrice qui modifiera l'image capturée. D'un autre côté, les modifications appliquées à n'importe quel MC est enregistré dans la matrice *movie_clip.transform.matrix*, il fallait donc traduire cette matrice pour que la capture suive le mouvement du viseur de la caméra, puis passer cette matrice en deuxième paramètre de la méthode *DRAW*.



Projet : Jeu pédagogique en Flash Étude Technique

```
56 function capture(nr) {
57     photo1_mc.sujet1._alpha = 0;
58     photo1_mc.sujet2._alpha = 0;
59     photo1_mc.sujet3._alpha = 0;
60     snapshot = new BitmapData(follow._width, follow._height);
61     var m:Matrix = photo1_mc.transform.matrix;
62     m.tx = -follow._x + 580;
63     m.ty = -follow._y + 423;
64     snapshot.draw(photo1_mc, m);
65     capture_mc.attachBitmap(snapshot, 1);
66 }
```

Conclusion

Nous avons fait certains choix techniques du à des contraintes, nous avons déjà trouvé la solution aux principaux problèmes rencontrés. Nous concluons donc quant à la faisabilité du projet.