12 Le Web en vrai relief

Introduction

L'impression de relief est créée par le cerveau. Il analyse les deux images reçues par chaque œil et reconstitue le volume en superposant ces deux images. Pour créer des images en relief, nous devons donc organiser l'affichage de sorte que le cerveau puisse lire deux sources d'images distinctes, dans un temps donné, le plus court possible.

Il existe différentes techniques de montage en relief. La plus accessible, tous supports confondus, demeure la technique de l'anaglyphe. L'anaglyphe se caractérise par la création d'une image composée de couches séparées : d'une part, le rouge et d'autre part, le vert et bleu. Le rouge est dissocié du vert et du bleu qui eux, restent superposés. Cette technique implique certes l'utilisation de lunettes rouges et bleues (ou rouges et vertes) pour l'interpréter, mais peut se déployer aussi bien sur un écran vidéo, informatique, cinéma, que sur le papier (exemples d'anaglyphes : **http://www.dailymotion.com/video/xb6ez7_visitez-paris-en-3d-16_creation**).

Les autres techniques demeurent plus complexes et sont généralement réservées à des dispositifs plus lourds à mettre en œuvre, mais qui répondent à une exigence manifeste sur le choix et la qualité des couleurs. Ainsi, sont également rencontrées :

- **La technique des lunettes passives.** Elle consiste à employer une paire de lunettes dont chaque verre affiche une trame soit horizontale soit verticale. L'image projetée est dissociée à travers l'utilisation de deux projecteurs. Chacun affiche une image pour une trame donnée, en reproduisant la trame correspondant à l'œil visé.
- **La technique des lunettes actives.** Elle consiste à placer un écran translucide à cristaux liquides devant chaque œil. Contrôlé par un système de laser qui les relie à l'écran, chaque verre alterne le passage de la lumière pour ne laisser passer que l'image qui correspond à l'œil visé. L'image est synchrone avec le rythme de balayage de la lunette. Ces lunettes requièrent une légère alimentation électrique généralement assumée par l'utilisation de piles. Cette technique est favorisée dans la production audiovisuelle car elle ne limite pas le choix des couleurs lors de la capture, contrairement à l'anaglyphe pour lequel ce qui est trop rouge ou trop bleu, du fait du principe de couches séparées, ne peut être traité en relief.
- **La technique du réseau lenticulaire.** Cette technique ne requiert pas de lunettes. Il s'agit de placer sur un écran une couche uniforme de lentilles convexes très fines, collées les unes à la suite des autres. L'image affichée est décomposée en fines trames à raison de 8 à 12 trames différentes par lentille. Il faut donc capturer 8 à 12 images pour reconstituer un volume. C'est en se déplaçant le long de l'écran lenticulaire que le volume seulement réapparaît, et à une distance bien définie. Ce dispositif complexe n'est utilisé que pour les images fixes dans des zones d'exposition à faible recul et pour des usagers en mouvement (vitrines, galeries ou salons).

Dans ce chapitre, nous abordons la création d'images en relief de type anaglyphe, avec Photoshop, puis, la gestion de l'affichage de ces images en ActionScript. Nous présentons, en fin de chapitre, un moteur de rendu en relief, entièrement dynamique, qui permet de convertir automatiquement en véritable objet en relief tout objet placé dans un espace 3D dans Flash.

Pour étudier les exemples de ce chapitre, il est recommandé de disposer de lunettes pour anaglyphes, à films rouge et bleu, disponibles dans votre kiosque, chez votre marchand de jouet ou sur Internet. Certains opticiens et photographes proposent également ce type d'équipement (voir aussi **http://www.alpes-stereo.com/lunettes.html**).

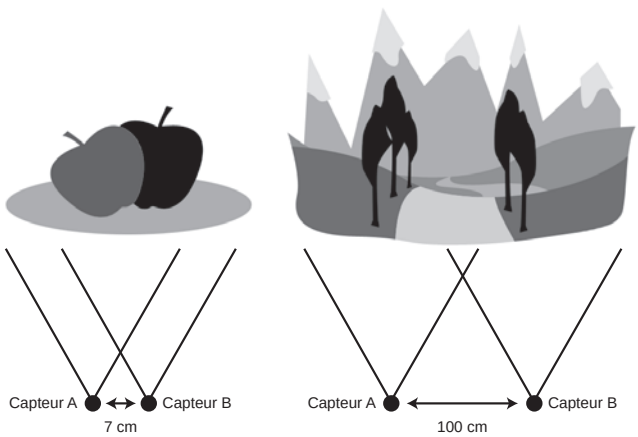
Prise de vues pour le relief

Pour créer un affichage en relief avec un effet le plus précis possible, nous devons prendre en compte un certain nombre d'éléments. Dans le cadre de ce chapitre, nous en évoquons seulement les grands principes. Pour connaître les techniques de création d'images en relief de manière détaillée, consultez le site de David Romeuf : **http://www.david-romeuf.fr**, très largement documenté sur le sujet.

Pour vous aider à considérer toutefois certains principes élémentaires, inhérents à la création d'images en relief, voici quelques recommandations :

- Vous devez d'abord disposer de deux images du même sujet, mais vous devez décaler le point de vue de chacune par rapport à l'autre. Vous reconstituez ainsi le point de vue des yeux qui est à l'origine de la vision en relief.
- Lors de la prise de vue photographique, la distance entre les deux points de vue, pour reconstituer un tel effet, doit être directement proportionnelle à la distance entre l'obturateur et le sujet visé. Ainsi, si vous photographiez un sujet de près, la distance entre les deux points de vue doit être égale à la distance entre deux pupilles d'œil humain (6 à 10 cm). Si vous capturez, en revanche, un paysage lointain (une montagne, une architecture, un volume paysagé ou urbain), vous devez séparer les points de vue de plusieurs dizaines de centimètres, voir d'environ un mètre pour les grands paysages (voir Figure 12.1).

Figure 12.1
Distance entre les points de prise de vue.



Dans la conception d'une image anaglyphe pour le relief, nous devons également éviter de capturer des objets à dominante colorée proche de celle de l'une des deux lentilles. Un objet de couleur rouge, par exemple, ne pourrait être perçu par l'une des deux lentilles et son volume ne pourrait donc pas être reconstitué. Évitez, par conséquent, la mise en scène d'objets rouges et bleus, si vous travaillez avec la technique de l'anaglyphe,

Pour réaliser un anaglyphe, nous décalons d'abord les couches de couleurs rouge, vert et bleu, dans Photoshop. Une fois l'image exportée pour le Web, nous pouvons en contrôler l'affichage, dans Flash, avec ActionScript.

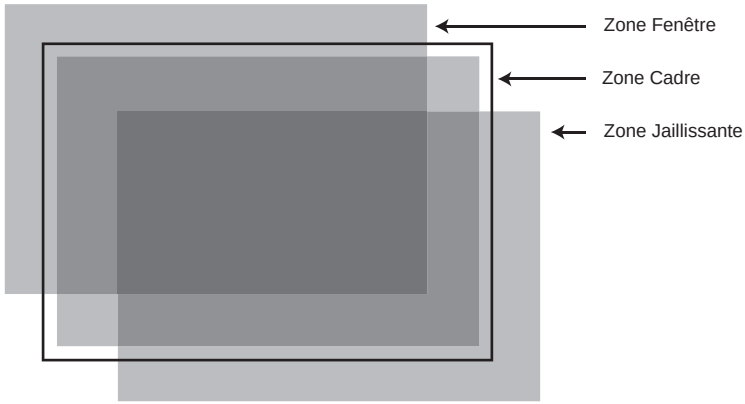
Réaliser un anaglyphe avec Photoshop

Dans cette section, nous allons d'abord comprendre le principe de l'image jaillissante ou fenêtre, à partir d'une image simple. Puis, nous verrons comment convertir deux prises de vue séparées, en un seul anaglyphe, à l'aide de Photoshop.

Le principe de l'image jaillissante ou fenêtre

Dans une image en relief, nous distinguons trois zones. La zone cadre, la zone jaillissante et la zone fenêtre (voir Figure 12.2).

Figure 12.2
Définition des zones d'affichage pour le relief.



La zone cadre représente la partie sans relief de l'image. La zone jaillissante est la partie qui se place devant le cadre, c'est-à-dire, la partie de l'image qui sort de l'écran, au premier plan. La partie fenêtre représente la partie située à l'arrière-plan du cadre, c'est-à-dire, la partie en retrait et située en profondeur de l'écran.

Le principe des trois zones est important, car selon la manière dont nous créons l'anaglyphe dans Photoshop, selon le type de décalage apporté à la couche de couleur rouge, nous pouvons choisir si l'image sera placée en retrait (fenêtre) ou en avant (jaillissante) (voir Figure 12.3 et 12.4).

Figure 12.3
Anaglyphe fenêtre.

Figure 12.4
Anaglyphe jaillissant.

Dans cette section, nous allons convertir une image initialement "plate", en image fenêtre, puis en image jaillissante. Il n'est pas nécessaire de disposer de deux images spécifiques pour le relief pour effectuer cette manipulation. Toute l'image sera simplement projetée dans son intégralité, soit vers l'arrière, soit vers l'avant. Sur ce principe, nous pouvons donc déjà imaginer pouvoir composer assez simplement, avec Photoshop, des images en relief à partir d'objets 2D détourés, sur des plans dissociés. Nous abordons cette technique un peu plus loin dans ce chapitre.

La zone fenêtre

Dans cette section, nous allons voir comment réaliser une image de type fenêtre.

Exemples > relief > Lacoste.jpg

1. Lancez Photoshop.
2. Ouvrez le document "Lacoste.jpg", situé dans le dossier "relief" des exemples du livre. Cette image est normale et n'affiche pas de relief. Elle est en RVB et son unique calque est aplati (voir Figure 12.5).

Figure 12.5
Aperçu de l'image Lacoste.jpg.

3. Affichez la fenêtre Couches en déroulant le menu Fenêtre > Couches (voir Figure 12.6).

Figure 12.6
Fenêtre Couches.

La fenêtre affiche un aperçu des composantes colorimétriques de l'image, à travers différentes couches RVB. La couche RVB affiche les composantes de l'image dans sa globalité. Les couches R, V et B affichent respectivement les couleurs rouge, vert et bleu de l'image. Vous pouvez cliquer sur chacune d'entre elles pour en distinguer les propriétés.

Nous allons déplacer la couche Rouge à gauche, d'une dizaine de pixels, pour faire sortir l'image de la zone cadre et créer un effet Fenêtre.

1. Cliquez sur la couche Rouge. Elle est active. Les autres couches sont masquées. Dans l'image, la luminosité des valeurs de rouge est matérialisée en noir et blanc (voir Figure 12.7).

Figure 12.7
Activation de la couche Rouge.

2. Pour déplacer la couche de 10 pixels vers la gauche, utilisez l'outil de déplacement (raccourci V). Sélectionnez tous les pixels de la couche en faisant Ctrl+A (Windows) ou Cmd+A (Mac).

3. Puis, appuyez simultanément sur la touche Maj+Flèche gauche du clavier. La couche Rouge est aussitôt déplacée de 10 pixels vers la gauche.

4. Désactivez la sélection en faisant Ctrl+D (Windows) ou Cmd+D (Mac).

5. Cliquez à nouveau sur la couche intitulée RVB pour afficher l'intégralité de l'image. L'image en couleur affiche désormais un décalage entre le rouge et les composantes vert et bleu. L'image obtenue est un anaglyphe (voir Figure 12.8).

Figure 12.8
Aperçu de l'anaglyphe.

6. Recadrez éventuellement l'image pour supprimer la marge située à droite.
7. Puis, observez l'image à l'aide de lunettes rouge et bleue, en prenant soin de placer le filtre rouge sur l'œil gauche.

Vous pouvez maintenant voir une image projetée à l'intérieur de l'écran, dans la zone fenêtre.

La zone jaillissante

Dans cette section, nous allons voir comment réaliser une image de type jaillissante.

Exemples > relief > Lacoste-chien.psd

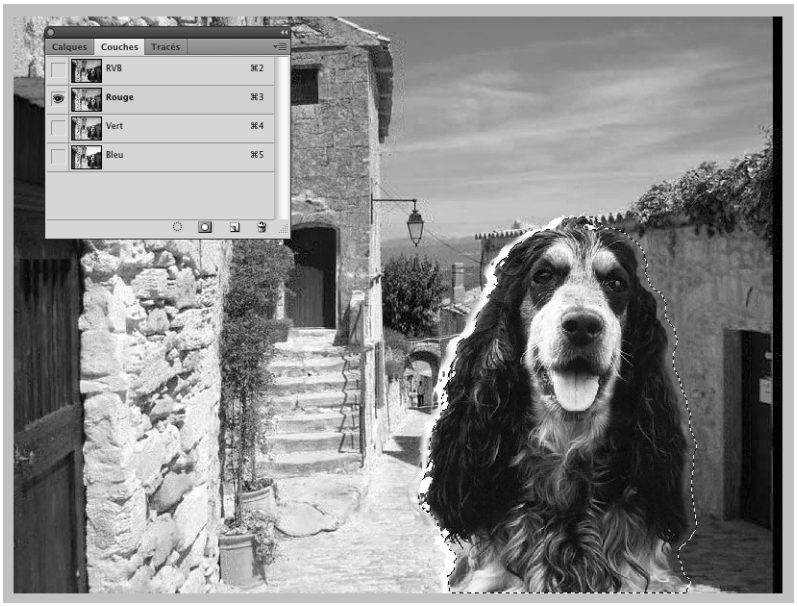
Le principe pour créer un volume jaillissant est identique à celui utilisé pour le mode Fenêtre, sauf que nous déplaçons la couche Rouge vers la droite afin de projeter l'image en premier plan. Notez cependant que, parce que notre cerveau les analyse ainsi, l'effet jaillissant ne fonctionne qu'avec des éléments qui représentent des formes en volume, et non en profondeur comme cette ruelle représentée ici. Pour que l'effet fonctionne en mode jaillissant, nous utilisons donc, dans notre exemple, un sujet détouré qui sera positionné au premier plan.

1. Ouvrez le fichier Lacoste-chien.psd (voir Figure 12.9).

Figure 12.9
Aperçu du document Lacoste-chien.psd.

2. Activez le calque du chien.
3. Directement, dans la fenêtre Couches, cliquez sur la couche Rouge.
4. Sélectionnez tous les pixels (Ctrl+A pour Windows ou Cmd+A pour Mac) et déplacez-les de 10 pixels vers la droite à l'aide de l'outil Flèche et déplacement (raccourci V). Vous remarquez que le déplacement génère une frange blanche de 10 pixels à gauche du sujet (voir Figure 12.10).

Figure 12.10
Frange à gauche du sujet.



Nous pouvons la supprimer en gommant directement sur toutes les couches du calque chien en même temps.

5. Intervertissez la sélection encore active en faisant Ctrl+Maj+i (Windows) ou Cmd+Maj+i (Mac).
6. Cliquez d'abord sur la couche RVB pour atteindre l'ensemble des composantes colorimétriques du calque.
7. Sélectionnez l'outil Gomme – raccourci E, comme *Erase* qui signifie gommer en anglais – (voir Figure 12.11).

Figure 12.11
Sélection de l'outil Gomme.

8. Puis, gommez la partie gauche du calque chien, matérialisée à présent en rouge (voir Figure 12.12).

Figure 12.12
Gommage de la frange à gauche.

9. Une fois la frange supprimée, désactivez la sélection en faisant Ctrl+D (Windows) ou Cmd+D (Mac).
10. Reprenez les lunettes rouge et bleue. Placez-vous à plus d'un mètre de votre écran. Le chien semble sortir de l'écran tandis que la ruelle se prolonge en profondeur (voir Figure 12.13).

Figure 12.13
Reconstitution du résultat obtenu.

Cette composition est affichée en relief, bien que chaque sujet ne soit pas capturé en relief. Seuls les plans sur lesquels sont disposés les objets sont en relief. Pour réaliser un véritable anaglyphe en relief, vous devez utiliser deux images qui représentent le même sujet, mais pris à quelques centimètres de décalage.

Créer l'anaglyphe à partir de deux images

Nous abordons ici la création de l'anaglyphe, dans Photoshop, à partir de deux images prises sur le même sujet avec un intervalle de 10 cm.

Exemples > relief > relief.psd

Ouvrez le document "relief.psd". La composition représente quelques fruits disposés autour d'un mug à thé. La fenêtre des calques (Fenêtre > Calques) affiche deux images. Le calque du haut représente une prise de vue effectuée au niveau de l'œil droit (voir Figure 12.14). Celui du bas représente le point de vue de l'œil gauche (voir Figure 12.15).

Figure 12.14
Aperçu du calque œil droit.

Figure 12.15
Aperçu du calque œil gauche.

Pour réaliser un anaglyphe à partir de deux images, nous plaçons les informations de la couche Rouge d'une première image à la place des informations de la couche Rouge de la seconde :

1. Affichez la fenêtre Couches et cliquez directement sur la couche Rouge. Puis, sélectionnez tous les pixels de la couche en faisant Ctrl+A sous Windows, ou Cmd+A sous Mac (voir Figure 12.16).

Figure 12.16
Sélection de la couche Rouge.

2. Faites Ctrl+C (Windows) ou Cmd+C (Mac) pour copier la couche Rouge.
3. Revenez dans la fenêtre des calques et activez maintenant l'autre calque"photo œil gauche". Pour voir ce calque placé, masquez le calque précédent, situé au sommet la fenêtre des calques (voir Figure 12.17).

Figure 12.17
Isolement du calque œil gauche.

4. Revenez dans la fenêtre Couches et cliquez sur la couche Rouge pour l'activer.
5. Sélectionnez toute la couche en faisant Ctrl+A (Windows) ou Cmd+A (Mac).
6. Puis, collez les informations Rouges du calque précédent en faisant Ctrl+V (Windows) ou Cmd+V (Mac).
7. Activez enfin la couche globale RVB de ce calque pour en visualiser le rendu (voir Figure 12.18).

   La nouvelle couche Rouge est bien appliquée au document. Mais elle est trop décalée par rapport à l'image de gauche. Pour que le relief prenne, nous devons rapprocher les éléments. À l'aide de l'outil de déplacement (raccourci V), déplacez la couche Rouge de 10 ou 20 pixels pour la recentrer (voir Figure 12.19).

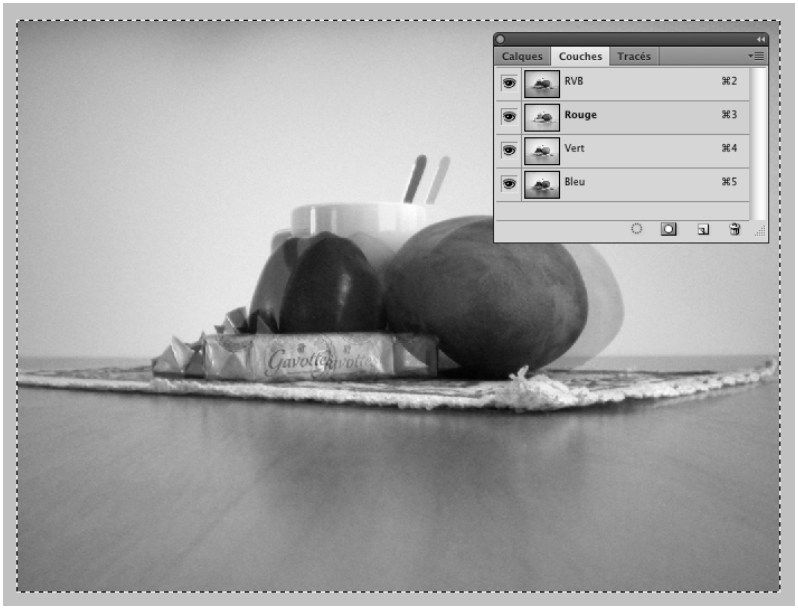


Figure 12.18
Placement de la nouvelle couche Rouge.

Figure 12.19
Recentrage de la couche Rouge.

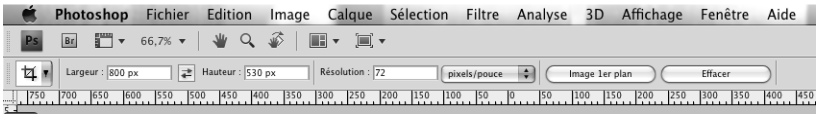
L'image est prête à être recadrée. Elle mesure 1 024 × 768 pixels. Nous allons la recadrer afin qu'elle occupe les mêmes dimensions que celles de notre document Flash :

8. Activez l'outil de recadrage – raccourci C, comme *Crop* qui signifie rogner en anglais – (voir Figure 12.20).

Figure 12.20
Sélection de l'outil de recadrage.

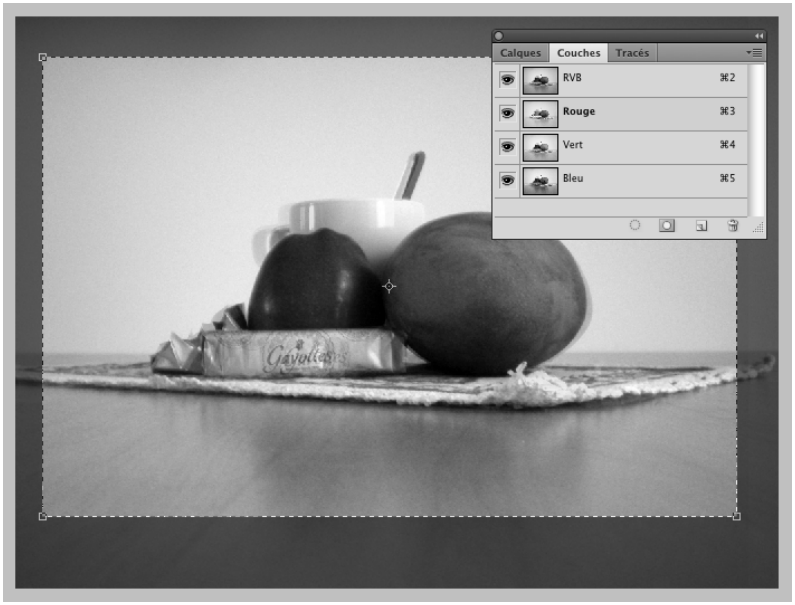
9. Dans les options, situées au sommet de l'interface de Photoshop, sous la barre de menu, inscrivez les valeurs : 800px, 530px et 72dpi (voir Figure 12.21).

Figure 12.21
Options de recadrage.



10. Recadrez l'image en centrant approximativement le sujet. Puis validez en cliquant sur Entrée (voir Figure 12.22).

Figure 12.22
Aperçu de la zone de recadrage.



11. Une fois l'image recadrée, vous pouvez aussi l'aplatir. Faites Calques > Aplatir l'image.
12. Puis exportez pour le Web en faisant Fichier > Enregistrer pour le Web et les périphériques. Choisissez une option de compression JPEG de qualité supérieure ou égale à 60 (voir Figure 12.23). Puis confirmez l'enregistrement.

Figure 12.23
Aperçu de la fenêtre d'enregistrement pour le Web.

L'image est prête à être importée dans Flash.

Gérer un anaglyphe en ActionScript

L'intégration d'un anaglyphe dans Flash est relativement simple puisqu'il suffit d'importer l'image dans la scène. Mais il convient aussi de bien situer le type d'image dont nous disposons selon le contexte d'exécution. Si vous affichez une image jaillissante, il faut naturellement la placer de préférence au sommet de la liste d'affichage ou sur un calque placé au-dessus des autres. Inversement, si l'image est de type fenêtre, vous la placerez à l'arrière des autres objets de la scène.

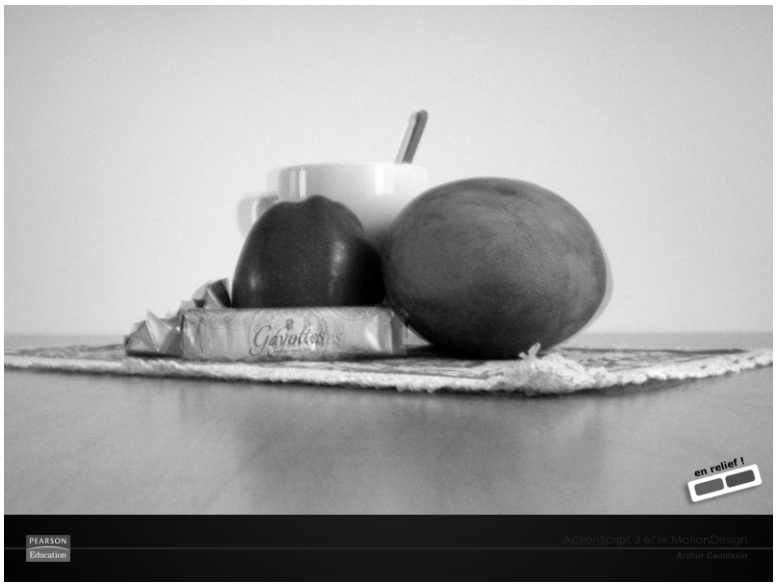
Cela signifie aussi que pour une image jaillissante, dans un contexte de scène en 3D, vous lui affecterez un index z paradoxalement inférieur à 0. Pour un type fenêtre, vous appliquerez un index z supérieur à 0. Dans Flash, nous rappelons le z augmente en allant vers le fond, tandis qu'il prend une valeur négative dès qu'il se rapproche de l'écran. Plus l'objet est proche de l'écran et au premier plan, plus son index z diminue.

Exemples > ch11_vraiRelief_1.fla

Dans le document "ch11_vrairelief_1.fla", sur la scène, nous pouvons voir un MovieClip qui contient deux images. Une image classique est placée sur l'image 1 du scénario. La version en relief est placée sur l'image 2. Au-dessus, sur la scène principale, un bouton en forme de lunette 3D permet de basculer de l'une à l'autre.

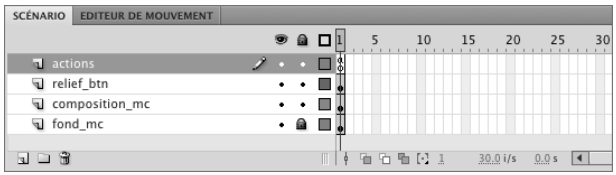
En publiant le document, quand nous cliquons sur les lunettes, l'image en relief remplace effectivement l'image normale (voir Figure 12.24).

Figure 12.24
Aperçu du document.



Dans la fenêtre de scénario, au-dessus du calque `Actions`, nous distinguons le MovieClip `composition_mc` du bouton `relief_btn` (voir Figure 12.25).

Figure 12.25
Aperçu de la fenêtre de scénario.



Dans la fenêtre Actions, apparaît le code suivant :

```
composition_mc.stop();
//
relief_btn.addEventListener(MouseEvent.CLICK,basculerEnRelief);
function basculerEnRelief (evt:MouseEvent) {
   if (composition_mc.currentFrame==1) {
      composition_mc.nextFrame();
   }else {
      composition_mc.prevFrame();
   }
}
```

Dans ce programme, nous plaçons un stop au début du MovieClip qui contient les deux images afin d'éviter qu'une animation ne se produise.

Plus bas, un écouteur, attaché au bouton `relief_btn`, lance le changement d'image en fonction de celle qui est active. L'image active est détectée, elle, grâce à la méthode `currentFrame()`.

Dans cette condition, si l'image active est la première, nous basculons vers la seconde avec la méthode `nextFrame()` qui désigne l'image suivante. Nous revenons, en revanche, à la première image si c'est la seconde qui est identifiée, avec la méthode `prevFrame()` qui désigne l'image précédente.

La vidéo en relief. *La technique du relief peut également être appliquée à un flux vidéo. Pour cela, vous devez disposer soit d'une composition vidéo à plusieurs calques et déplacer la couche Rouge de chaque calque à gauche ou à droite en fonction de l'effet souhaité (avec After Effects ou Motion par exemple). Soit vous devez partir de deux flux vidéo distincts, représentant le point de visé des deux yeux, comme pour des images fixes, et fusionner le tout en un seul flux en prenant la couche rouge de l'un que vous placez à la place de la couche rouge de l'autre.*

Dans After Effects, utilisez l'effet Perspective > Lunette 3D pour regrouper les flux vidéo. Dans les réglages d'effets, source G désigne le point de vue de l'œil gauche, source D, le point de vue de l'œil droit. Balance couleur Rouge Bleu signifie vision 3D. Ajustez le Décalage de convergence (normalement, si les vidéos sont bien tournées, la valeur est proche de zéro). Ajustez la Balance en fonction de la densité des lunettes (mesurable chez votre opticien). Puis, exportez la vidéo obtenue en F4V ou en FLV et intégrez-la simplement avec un composant `FLVPlayBack`*, comme vu au Chapitre 7.*

À retenir

- *Pour créer une image en relief, nous devons reproduire la vision humaine en capturant deux images dont les points de visé sont distants, l'un de l'autre, d'environ 6,5 cm pour un sujet proche, à près d'1 mètre pour un grand paysage.*
- *Nous devons copier la couche Rouge d'une image pour remplacer celle de l'autre.*
- *L'image devient de type fenêtre dès lors que la couche Rouge est située à gauche.*
- *L'image devient jaillissante dès que la couche Rouge est placée sur la droite.*
- *Il est possible de réaliser des images en relief à partir d'images classiques. Pour cela, nous détourons chaque sujet et décalons la position de sa couche par rapport aux couches Rouge des autres calques de la composition.*
- *Il est possible, en ActionScript, de détecter la position courante d'une image de scénario, grâce à la méthode* `currentTarget()`.

Interface SWF en relief dynamique

Nous venons de voir comment se constitue un contenu en relief. Au chapitre précédent, nous avons vu comment agir sur les couches RVB et Alpha d'une image, dynamiquement. Précédemment encore, nous avons abordé la gestion de la 3D avec l'index z. En combinant toutes ces techniques, nous pouvons construire une interface 3D en relief dynamique.

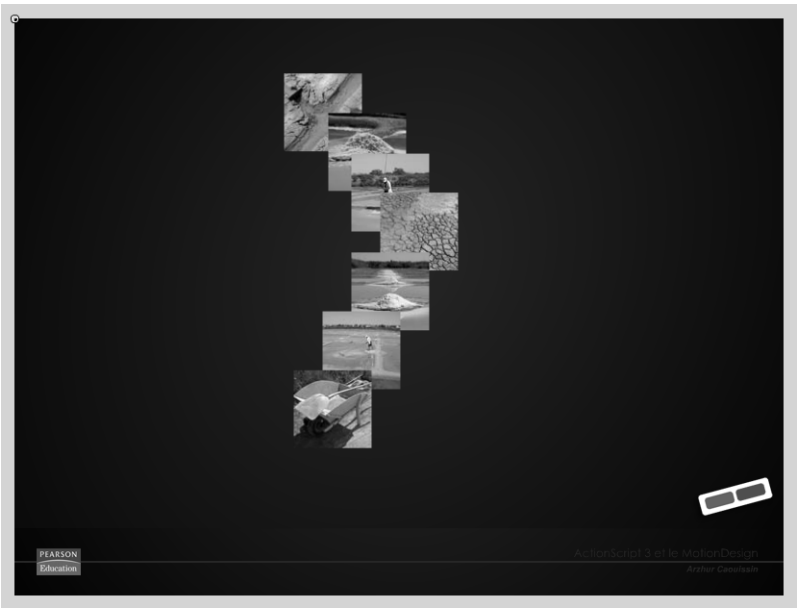
Le principe de l'interface en relief dynamique est de permettre le positionnement d'objets (textes, images, formes graphiques) dans un conteneur et de leur attribuer, pour chacun d'entre eux, un index z. En publiant le document, les couches Rouge, Vert et Bleu sont automatiquement séparées en fonction de la profondeur de z.

Dans cette section, nous allons étudier le mécanisme d'un moteur de rendu d'objets en vrai relief.

Exemples > ch12_vraiRelief_2.fla

Dans le document "ch12_vrairelief_2.fla", sur la scène principale, apparaît un MovieClip `composition_mc`, à l'intérieur duquel sont positionnés des symboles. Ces symboles sont répartis de sorte que celui qui doit apparaître au premier plan est situé naturellement au-dessus des autres. Dans le code du calque `Actions`, que nous allons voir, un index z est également attribué pour chacun d'entre eux (voir Figure 12.26). N'oubliez pas que pour qu'un objet puisse disposer d'un index z, celui-ci doit être un MovieClip.

Figure 12.26
Aperçu de la scène principale.



Lorsque le document est publié, si le bouton `relief_btn` situé à droite de l'écran est activé, la composition bascule presque instantanément en vrai relief. Le décalage de la couche Rouge est directement proportionnel à la position de l'objet sur l'index z (voir Figure 12.27).

Dans la fenêtre de scénario du document, au-dessus du calque `fond_mc`, sont répartis : le symbole `composition_mc`, qui affiche tous les éléments à projeter en relief ; au-dessus, le symbole `zoneDepotRelief_mc`, qui contient les objets utilisés par le moteur de relief ; ainsi que le bouton `relief_btn` (voir Figure 12.28).

Figure 12.27
Aperçu du document publié, avec l'option relief activée.

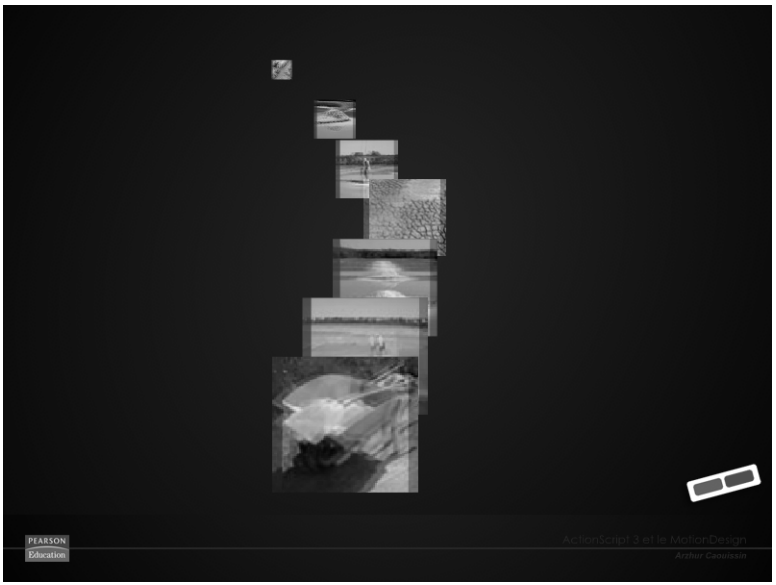
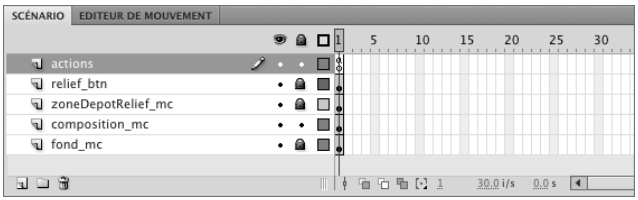


Figure 12.28
Scénario de la scène principale.



Dans la fenêtre Actions, nous pouvons lire le code suivant :

```
//------------------- personnalisation

var echelleZ:Number=400;
var echelleRelief:Number=0.015;
//
function indexZ() {
   // toujours <0
   composition_mc.p1_mc.z=-700;
   composition_mc.p2_mc.z=-600;
   composition_mc.p3_mc.z=-500;
   composition_mc.p4_mc.z=-400;
   composition_mc.p5_mc.z=-300;
   composition_mc.p6_mc.z=-200;
   composition_mc.p7_mc.z=-100;
}

relief_btn.addEventListener(MouseEvent.CLICK,basculerEnRelief);
function basculerEnRelief(evt:MouseEvent) {
   composition_mc.visible=false;
   moteurRelief();
}
```

```
//
//////////////////////////////////////////////////////////////////////////////////
//////////////////// MOTEUR RELIEF (Arzhur CAOUISSIN) ////////////////////
//
//////////////////////////////////////////////////////////////////////////////////

// tableauRouge
var tableauRouge:Array = new Array();
var filtreRouge:ColorMatrixFilter=new ColorMatrixFilter(tableauRouge);
tableauRouge = new Array();
tableauRouge=tableauRouge.concat([-1,0,0,0,255]);
tableauRouge=tableauRouge.concat([0,-1,0,0,0]);
tableauRouge=tableauRouge.concat([0,0,-1,0,0]);
tableauRouge=tableauRouge.concat([0,0,0,1,0]);
filtreRouge=new ColorMatrixFilter(tableauRouge);
// tableauVert
var tableauVert:Array = new Array();
var filtreVert:ColorMatrixFilter=new ColorMatrixFilter(tableauVert);
tableauVert = new Array();
tableauVert=tableauVert.concat([-1,0,0,0,0]);
tableauVert=tableauVert.concat([0,-1,0,0,255]);
tableauVert=tableauVert.concat([0,0,-1,0,0]);
tableauVert=tableauVert.concat([0,0,0,1,0]);
filtreVert=new ColorMatrixFilter(tableauVert);
// tableauBleu
var tableauBleu:Array = new Array();
var filtreBleu:ColorMatrixFilter=new ColorMatrixFilter(tableauBleu);
tableauBleu = new Array();
tableauBleu=tableauBleu.concat([-1,0,0,0,0]);
tableauBleu=tableauBleu.concat([0,-1,0,0,0]);
tableauBleu=tableauBleu.concat([0,0,-1,0,255]);
tableauBleu=tableauBleu.concat([0,0,0,1,0]);
filtreBleu=new ColorMatrixFilter(tableauBleu);
// tableauNoir
var tableauNoir:Array = new Array();
var filtreNoir:ColorMatrixFilter=new ColorMatrixFilter(tableauNoir);
tableauNoir = new Array();
tableauNoir=tableauNoir.concat([-1,0,0,0,0]);
tableauNoir=tableauNoir.concat([0,-1,0,0,0]);
tableauNoir=tableauNoir.concat([0,0,-1,0,0]);
tableauNoir=tableauNoir.concat([0,0,0,1,0]);
filtreNoir=new ColorMatrixFilter(tableauNoir);
//++++++++++++++++++++++++++++++++++++++

var largeurDonnees:Array=new Array();
var hauteurDonnees:Array=new Array();

function moteurRelief() {
   for (var i:Number=0; i<composition_mc.numChildren; i++) {
      largeurDonnees.push(composition_mc.getChildAt(i).width);
      hauteurDonnees.push(composition_mc.getChildAt(i).height);
   //
   if (i==composition_mc.numChildren-1) {
      var boucle:Timer=new Timer(500,1);
      boucle.addEventListener(TimerEvent.TIMER,lancerBoucle);
      boucle.start();
      indexZ();
```

```
        }
    }
}

function lancerBoucle(evt:TimerEvent) {
    affichageRelief();
}

function affichageRelief() {
    for (var i:Number=0; i<composition_mc.numChildren; i++) {
        //
        var enveloppeImage:BitmapData=new BitmapData(800,600,false,0x00000000);
        enveloppeImage.draw(composition_mc.getChildAt(i));
        var pixelsImage:Bitmap=new Bitmap(enveloppeImage);
        var capture:BitmapData=pixelsImage.bitmapData;
        //
        var bitmapR:BitmapData=new BitmapData(largeurDonnees[i],
        ➥ hauteurDonnees[i],true,0x000000);
        var bitmapV:BitmapData=new BitmapData(largeurDonnees[i],
        ➥ hauteurDonnees[i],true,0x000000);
        var bitmapB:BitmapData=new BitmapData(largeurDonnees[i],
        ➥ hauteurDonnees[i],true,0x000000);
        var bitmapA:BitmapData=new BitmapData(largeurDonnees[i],
        ➥ hauteurDonnees[i],true,0x000000);
        //
        var imageR:Bitmap=new Bitmap(bitmapR);
        var imageV:Bitmap=new Bitmap(bitmapV);
        var imageB:Bitmap=new Bitmap(bitmapB);
        var imageA:Bitmap=new Bitmap(bitmapA);
        //
        bitmapR.copyChannel(capture, capture.rect, new Point(0,0),
        ➥ BitmapDataChannel.RED, BitmapDataChannel.ALPHA);
        bitmapV.copyChannel(capture, capture.rect, new Point(0,0),
        ➥ BitmapDataChannel.GREEN, BitmapDataChannel.ALPHA);
        bitmapB.copyChannel(capture, capture.rect, new Point(0,0),
        ➥ BitmapDataChannel.BLUE, BitmapDataChannel.ALPHA);
        bitmapA.copyChannel(capture, capture.rect, new Point(0,0),
        ➥ BitmapDataChannel.ALPHA, BitmapDataChannel.ALPHA);
        //
        imageR.filters=[filtreRouge];
        imageV.filters=[filtreVert];
        imageB.filters=[filtreBleu];
        imageA.filters=[filtreNoir];
        //
        var AffichageRelief:MovieClip=new ZoneAffichageRelief();
        zoneDepotRelief_mc.addChildAt(AffichageRelief,i);
        // la couche RVB
        AffichageRelief.coucheAlpha_mc.addChild(imageA);
        imageA.x=composition_mc.getChildAt(i).x;
        imageA.y=composition_mc.getChildAt(i).y;
        // la couche Rouge
        AffichageRelief.coucheRouge_mc.addChild(imageR);
        imageR.x=composition_mc.getChildAt(i).x+(composition_mc.getChildAt(i).z)
        ➥ *echelleRelief;
```

```
            imageR.y=composition_mc.getChildAt(i).y;
            // la couche Bleu
            AffichageRelief.coucheVerte_mc.addChild(imageV);
            imageV.x=composition_mc.getChildAt(i).x;
            imageV.y=composition_mc.getChildAt(i).y;
            // la couche Bleu
            AffichageRelief.coucheBleue_mc.addChild(imageB);
            imageB.x=composition_mc.getChildAt(i).x;
            imageB.y=composition_mc.getChildAt(i).y;
            //
            imageR.scaleX=(composition_mc.getChildAt(i).z)/echelleZ;
            imageR.scaleY=(composition_mc.getChildAt(i).z)/echelleZ;
            imageV.scaleX=(composition_mc.getChildAt(i).z)/echelleZ;
            imageV.scaleY=(composition_mc.getChildAt(i).z)/echelleZ;
            imageB.scaleX=(composition_mc.getChildAt(i).z)/echelleZ;
            imageB.scaleY=(composition_mc.getChildAt(i).z)/echelleZ;
            imageA.scaleX=(composition_mc.getChildAt(i).z)/echelleZ;
            imageA.scaleY=(composition_mc.getChildAt(i).z)/echelleZ;
        }
    }
    //+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++//
    //+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++//
    //
    +++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++//
```

Le système est scindé en deux parties. La première donne accès aux éléments personnalisables directement en rapport avec les objets disposés dans la scène. La seconde constitue le moteur de rendu en relief.

Dans la première partie, nous définissons d'abord les cœfficients de déformation utilisés par le moteur :

```
var echelleZ:Number=400;
var echelleRelief:Number=0.015;
```

Le moteur fonctionne de la manière suivante. Nous plaçons des objets dans l'espace 3D à l'intérieur du symbole `composition_mc`. L'image projetée par ce conteneur est alors copiée en ActionScript, puis traitée, avant d'être redistribuée sur un autre conteneur : `zoneDepotRelief_mc`. Pendant le traitement, les couches sont séparées et l'une d'entre elles, la Rouge, est décalée.

Une fois le traitement terminé, les couches sont distribuées dans un MovieClip placé dynamiquement en ActionScript, à partir d'une occurrence exportée de la bibliothèque. Cette occurrence contient quatre MovieClip destinés respectivement pour chacune des quatre composantes de couleur Rouge, Vert, Bleu et Alpha.

Afin que les couches colorimétriques fusionnent correctement, nous avons appliqué, sur les deux premiers MovieClip, une propriété d'affichage de type Ajout (c'est un mode de fusion accessible depuis l'inspecteur de propriétés). Ainsi, lorsque les trois couches RVB sont superposées, sans décalage, l'image est entièrement reconstituée. Mais, pour reconstituer l'effet de relief, nous conservons naturellement le décalage de la couche Rouge.

Le symbole `AffichageRelief`, disponible dans la bibliothèque, est exporté pour ActionScript avec le nom de classe `ZoneAffichageRelief`. C'est lui qui sera distribué dans le symbole `zoneDepotRelief_mc`, autant de fois que d'objets à reconstituer.

Le langage

Exporter un symbole pour ActionScript

Pour exporter un symbole pour ActionScript en vu de réaliser un interfaçage dynamique, dans la bibliothèque (Ctrl+L pour Windows ou Cmd+L pour Mac), directement sur l'objet à exporter, faites clic-droit > Propriétés. Puis, dans la boîte de dialogue, cochez la case Exporter pour ActionScript. Attribuez ensuite un nom de classe à l'objet sans caractère spécial, ni accent, espace ou signe particulier. En validant, le player fabriquera pour nous cette classe à la volée si celle-ci n'existe pas.

La variable `echelleZ` permet de retranscrire l'échelle de déformation des couches, selon l'index z attribué à chaque objet :

```
var echelleZ:Number=400;
```

Ce cœfficient est utilisé pour recréer la profondeur, car en réalité, dans notre programme, les propriétés 3D que nous avons affectées aux objets de la scène ne sont pas prises en compte lorsque le calcul d'affichage en relief est appliqué. Elle n'est appliquée qu'une fois les couches séparées. Les propriétés `width` et `height` utilisées, perçoivent, en effet, des valeurs différentes lorsque deux couches alpha de mêmes dimensions sont placées à un index z distinct. Une incohérence que nous contournons en activant l'affichage 3D uniquement lorsque le calcul de séparation des couches est terminé. Sans quoi les couches alpha de chaque objet auraient été bien plus grandes que les objets eux-mêmes.

La deuxième variable, `echelleRelief`, est employée pour définir la valeur de décalage de la couche Rouge :

```
var echelleRelief:Number=0.015;
```

Plus cette valeur est élevée, plus le décalage de la couche Rouge sera marqué.

À la suite, une fonction détermine l'index z de chaque objet placé dans le symbole `composition_mc` :

```
function indexZ() {
   // toujours <0
   composition_mc.p1_mc.z=-700;
   composition_mc.p2_mc.z=-600;
   composition_mc.p3_mc.z=-500;
   composition_mc.p4_mc.z=-400;
   composition_mc.p5_mc.z=-300;
   composition_mc.p6_mc.z=-200;
   composition_mc.p7_mc.z=-100;
}
```

Le premier objet de la liste d'affichage, celui placé en premier plan dans le symbole `composition_mc`, affiche un index le plus proche de l'écran (–700). Les autres se rapprochent progressivement de la valeur 0. Ces valeurs sont isolées dans une fonction de manière à permettre au moteur de capturer les dimensions des objets avant de modifier leur index z, juste avant de lancer le mode d'affichage en relief. Afin de préserver les proportions d'échelle des objets en fonction de leur index z, nous limitons la gestion des objets à un index inférieur à 0, et donc, à un relief jaillissant.

Ensuite, un gestionnaire d'événements appelle le moteur de rendu, sur action de l'utilisateur :

```
relief_btn.addEventListener(MouseEvent.CLICK,basculerEnRelief);
function basculerEnRelief(evt:MouseEvent) {
   composition_mc.visible=false;
   moteurRelief();
}
```

Nous masquons d'abord le symbole `composition_mc` qui affiche les objets dans un espace 2D. Puis, nous activons la fonction `moteurRelief()` qui va placer, dans le clip vide `zoneDepotrelief_mc`, situé au premier plan, les objets d'affichage en relief.

Intervient ensuite le moteur de rendu. Le moteur génère, pour commencer, autant de filtres colorimétriques que de couches à isoler. Dans ce système, nous utilisons quatre couches. La couche Rouge permet de reconstituer le relief en le décalant d'une valeur proportionnelle à l'index z des objets. Les couches Vert et le Bleu sont séparées, mais resteront superposées pour recréer la base de l'image. L'Alpha, enfin, va permettre d'éviter que les couches, même superposées, apparaissent légèrement translucides. Souvenez-vous, nous avons appliqué un mode de fusion par Ajout pour les couches Roug et Vert. Or, la couche Bleu n'est pas nécessairement entièrement opaque. Cela dépend de l'image. Si nous n'ajoutons pas une couche supplémentaire qui reprenne l'ensemble des informations opaques de l'image, nous obtiendrons des images partiellement translucides.

Dans un premier temps, nous définissons donc les filtres de couleurs que nous appliquons plus loin, à chaque couche séparée :

```
// tableauRouge
var tableauRouge:Array = new Array();
var filtreRouge:ColorMatrixFilter=new ColorMatrixFilter(tableauRouge);
tableauRouge = new Array();
tableauRouge=tableauRouge.concat([-1,0,0,0,255]);
tableauRouge=tableauRouge.concat([0,-1,0,0,0]);
tableauRouge=tableauRouge.concat([0,0,-1,0,0]);
tableauRouge=tableauRouge.concat([0,0,0,1,0]);
filtreRouge=new ColorMatrixFilter(tableauRouge);
```

Nous déclinons le principe pour chaque couche. À la suite, nous créons un tableau (`Array`) qui enregistre les dimensions des images, avant qu'elles ne soient projetées en relief. Le premier tableau stocke les largeurs et le second, les hauteurs :

```
var largeurDonnees:Array=new Array();
var hauteurDonnees:Array=new Array();
```

Mécanisme d'un tableau (`Array`). *Rappel : un tableau fonctionne un peu comme la définition d'une variable, à ceci près que la structure peut implémenter un nombre de valeurs indéterminées. Pour ajouter une valeur dans un tableau, nous employons la méthode `push()`. En paramètre de cette méthode, nous spécifions la valeur à ajouter.*

Dans notre exemple, les deux tableaux sont instanciés en amont. C'est à travers une boucle `for` que nous récupérons les valeurs. Cette boucle permet d'exécuter autant d'enregistrements que le symbole `composition_mc` affiche d'objets (en fonction de `numChrildren`) :

```
function moteurRelief() {
   for (var i:Number=0; i<composition_mc.numChildren; i++) {
```

```
            largeurDonnees.push(composition_mc.getChildAt(i).width);
            hauteurDonnees.push(composition_mc.getChildAt(i).height);
            //
            if (i==composition_mc.numChildren-1) {
               var boucle:Timer=new Timer(500,1);
               boucle.addEventListener(TimerEvent.TIMER,lancerBoucle);
               boucle.start();
               indexZ();
            }
         }
      }

      function lancerBoucle(evt:TimerEvent) {
         affichageRelief();
      }
```

Dès que la boucle est terminée (`i==composition_mc.numChildren-1`), nous activons le moteur de rendu avec l'appel de la fonction `affichageRelief()`. Dans ce dispositif, nous devons attendre que la boucle soit terminée, car le moteur utilise les données enregistrées par la boucle pour effectuer le calcul d'affichage. Si nous lancions le moteur avant d'avoir défini les valeurs nécessaires, celui-ci ne pourrait fonctionner et retournerait une erreur. Pour garantir le délai, nous ajoutons même un chronomètre qui active la fonction seulement 500 millisecondes après l'enregistrement, soit une demi-seconde après l'enregistrement. Pour nous assurer que le chronomètre se déclenche uniquement suite aux instructions lancées en début de boucle, nous le plaçons à l'intérieur de la boucle.

À l'intérieur du moteur de rendu (fonction `affichageRelief`), plus bas dans le code, une autre boucle `for`, de même structure, instancie autant d'images que d'objets dans le conteneur :

```
var enveloppeImage:BitmapData=new BitmapData(800,600,false,0x00000000);
enveloppeImage.draw(composition_mc.getChildAt(i));
```

La méthode `BitmapData()` génère d'abord une surface d'affichage à quatre canaux (Alpha, Rouge, Vert et Bleu). Puis, la méthode `draw()` y place une capture de l'objet appelé en paramètre. La méthode `draw()` dessine, pour ainsi dire, sur l'objet `BitmapData`, ce qui est appelé en paramètre de la méthode `draw()`.

Ensuite, nous définissons chaque image capturée en tant qu'objet, afin de permettre de les redistribuer ensuite dans d'autres conteneurs et de leur appliquer des filtres et des propriétés :

```
var imageR:Bitmap=new Bitmap(bitmapR);
var imageV:Bitmap=new Bitmap(bitmapV);
var imageB:Bitmap=new Bitmap(bitmapB);
var imageA:Bitmap=new Bitmap(bitmapA);
```

À travers la méthode `copyChannel()`, nous récupérons la couche correspondant à une des quatre composantes colorimétriques. C'est, en somme, le noyau dur du moteur que nous définissons là :

```
bitmapR.copyChannel(capture, capture.rect, new Point(0,0),
BitmapDataChannel.RED, BitmapDataChannel.ALPHA);
```

En paramètre de cette méthode, nous désignons, respectivement :

- L'objet à partir duquel nous voulons extraire une couche (`capture`).
- La surface que nous voulons extraire (c'est toujours un rectangle, ici, la propriété `rect` lit la surface de l'image avec `capture.rect`).
- Les coordonnées `x` et `y` du point d'origine de cette zone rectangulaire ; la couche composite concernée (RED, GREEN, BLUE ou ALPHA).
- La propriété opaque ou transparente de l'image. Dans le cas d'une transparence, elle est définie avec la propriété `BitmapDataChannel.ALPHA`).

Puis, nous appliquons les filtres définis en amont, sur chacune de ces couches, en fonction de leur teinte respective :

```
imageR.filters=[filtreRouge];
imageV.filters=[filtreVert];
imageB.filters=[filtreBleu];
imageA.filters=[filtreNoir];
```

Grâce à ces filtres, les couches qui étaient extraites (par défaut noires) adoptent à présent la teinte qui correspond à leur composante colorimétrique réelle.

Pour chaque objet, nous plaçons dynamiquement, dans le symbole `zoneDepotRelief_mc`, une instance du symbole `ZoneAffichageRelief` disponible dans la bibliothèque et exporté pour ActionScript :

```
var AffichageRelief:MovieClip=new ZoneAffichageRelief();
zoneDepotRelief_mc.addChildAt(AffichageRelief,i);
```

Pour chaque occurrence, nous l'ajoutons également à la liste d'affichage dans l'ordre qui correspond à l'empilement déjà défini pour le symbole `composition_mc` (`addChildAt(AffichageRelief,i)`).

Plus bas, nous affichons les occurrences dans le conteneur `zoneDepotRelief_mc`, en les distribuant en fonction de la couche véhiculée. Les couches Rouge et Vert sont placées dans un conteneur avec le mode de fusion par Ajout déjà actif. La couche Bleu est placée derrière. La couche Alpha, elle, se retrouve en dessous de toutes les autres et préserve l'opacité de chaque objet (voir Figure 12.29) :

```
// la couche RVB
AffichageRelief.coucheAlpha_mc.addChild(imageA);
imageA.x=composition_mc.getChildAt(i).x;
imageA.y=composition_mc.getChildAt(i).y;
```

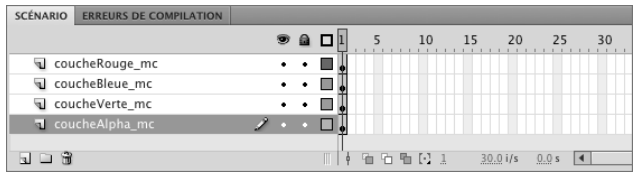


Figure 12.29
Ordre d'affichage des couches Alpha, Rouge, Vert et Bleu.

Nous en profitons pour ajuster aussi la position en X, principalement, pour la couche Rouge. Pour que l'effet de relief puisse être visible, nous devons effectivement décaler

l'abscisse du Rouge par rapport aux autres couches. C'est donc là que nous utilisons la variable `echelleRelief` qui nous aide à déterminer l'ampleur de ce décalage :

```
// la couche Rouge
AffichageRelief.coucheRouge_mc.addChild(imageR);
imageR.x=composition_mc.getChildAt(i).x+(composition_mc.getChildAt(i).z)
➥ *echelleRelief;
imageR.y=composition_mc.getChildAt(i).y;
```

La position X de la couche Rouge est ainsi proportionnelle à la valeur du `x` de chaque objet.

Nous terminons la fonction en modifiant l'échelle `scaleX` et `scaleY` de chaque objet en fonction de son index `x`, pour recréer la déformation induite par la mise en relief des objets dans l'espace 3D. Rappelez-vous que l'affichage du rendu est déterminé à partir d'objets 2D, placés dans `composition_mc`, et, à ce titre, nous oblige à déformer le tout en fonction de leur profondeur, que nous avons attribuée ailleurs, pour conserver l'illusion de cette profondeur :

```
imageR.scaleX=(composition_mc.getChildAt(i).z)/-echelleZ;
imageR.scaleY=(composition_mc.getChildAt(i).z)/-echelleZ;
```

La variable `echelleZ`, également définie en amont du programme, permet de gérer la déformation pour obtenir un effet plus ou moins accentué de la perspective. Une valeur élevée diminue la déformation et donc, éloigne les points de fuite.

À retenir

- *Il est possible d'extraire une couche de composante colorimétrique à l'aide de la méthode* `copyChannel()`.
- *Il est possible réaliser un site en relief dynamique à partir de propriétés distinctes de contenus 2D, s'ils sont répartis par exemple sur un index x. Le moteur de rendu en relief sépare les couches RVBA et les redistribue alors en fonction de la profondeur x des objets.*
- *L'ordre d'affichage des objets dans la composition est déterminant pour le réalisme du dispositif en relief.*

Synthèse

Dans ce chapitre, vous avez vu comment créer des images pour le relief avec la technique des anaglyphes. Vous avez appris à les intégrer dans un document Flash. En décortiquant le mécanisme de fonctionnement de l'affichage en relief, et à l'aide des techniques abordées dans les précédents chapitres, vous avez également appris à créer un système d'affichage dynamique qui permette de convertir, automatiquement, tout objet situé dans un espace 2D ou 3D, en relief. Vous êtes en mesure de développer des systèmes d'affichage pour tout type de contenu dont la volumétrie apporte du sens (architecture, bijoux, œuvres d'art, design d'objets, produits high-tech, musées).