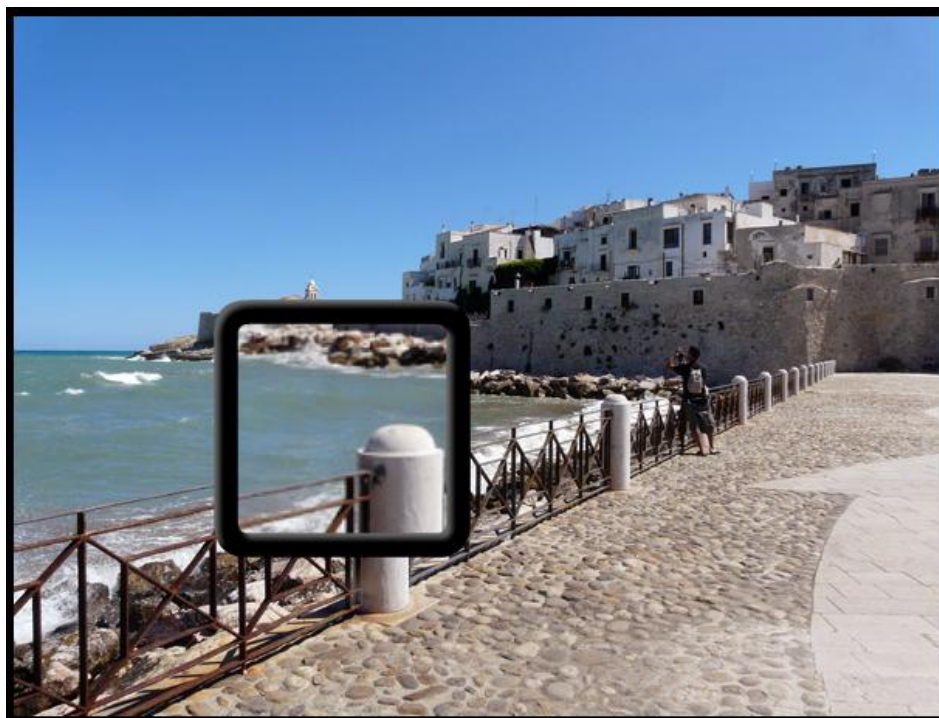


## Ćwiczenie dodatkowe 1

### Lupa powiększająca z filtrem mapa przemieszczeń

W tym ćwiczeniu stworzymy efekt lupy powiększającej.



Pobierz przykład (<http://jsekulska.kis.p.lodz.pl/studia.htm>).

Zademonstrujemy wykorzystanie filtrów obiektów wyświetlania na przykładzie klasy **Filtr Mapa przemieszczeń** (`displacementMapFilter`).

**Filtr Mapa przemieszczeń** jest dostępny wyłącznie w języku ActionScript 3.0. ActionScript 3.0 obejmuje dziesięć klas takich filtrów, które można nakładać na obiekty wyświetlania lub obiekty BitmapData. Każdy z filtrów, niezależnie od tego, czy jest to filtr prosty, czy złożony, można dostosować, korzystając z oferowanych przez niego właściwości. W ogólnym przypadku użytkownik ma do dyspozycji dwa ustawienia właściwości filtra. Wszystkie omawiane filtry umożliwiają ustawienie właściwości przez przekazanie wartości parametru do konstruktora obiektu filtra. Niezależnie od tego, czy użytkownik ustawia właściwości filtra przez przekazanie parametrów, możliwe jest również dostosowanie filtrów w późniejszym czasie przez ustawienie ich wartości dla właściwości obiektu filtra.

Klasa **DisplacementMapFilter** wykorzystuje wartości pikseli z obiektu BitmapData (znanego jako obraz mapy przemieszczeń) w celu uzyskania efektu przemieszczenia na nowym obiekcie. Obraz stanowiący mapę przemieszczeń jest zazwyczaj obiektem innym niż dany obiekt wyświetlany czy instancja BitmapData, na którą nakładany jest filtr.

Efekt przemieszczenia obejmuje przemieszczanie pikseli w obrazie filtrowanym — innymi słowy, w pewnym sensie przesuwanie ich z położenia oryginalnego do innego położenia. Filtru tego można użyć do wywołania efektu przesunięcia, wypaczenia lub nakrapiania.

Lokalizacja i wielkość przemieszczenia nakładana na dany piksel jest określona wartością koloru na obrazie mapy przemieszczeń. Podczas pracy z filtrami, poza określeniem obrazu stanowiącego mapę koloru użytkownik wskazuje również następujące wartości sterujące obliczaniem przemieszczenia z obrazu mapy:

- *Map point*: miejsce na obrazie filtrowanym, w którym zostanie nałożony górny lewy róg filtra przemieszczeń. Opcji tej można użyć, jeśli chce się tylko zastosować filtr do części obrazu.
- *X component*: określa, który kanał koloru obrazu stanowiącego mapę wpływa na położenie x pikseli.
- *Y component*: określa, który kanał koloru obrazu stanowiącego mapę wpływa na położenie y pikseli.
- *X scale*: wartość mnożnika określająca siłę przesunięcia wzdłuż osi x.
- *Y scale*: wartość mnożnika określająca siłę przesunięcia wzdłuż osi y.

- *Filter mode*: określa, jak program Flash Player lub AIR powinny wypełniać puste przestrzenie utworzone w wyniku przesunięcia pikseli w inne miejsce. Opcje zdefiniowane jako stałe w klasie `DisplacementMapFilterMode` powinny powodować wyświetlanie oryginalnych pikseli (tryb filtra `IGNORE`), powodować „zawijanie” pikseli znajdujących się w sąsiedztwie z drugiej strony obrazu (tryb filtra `WRAP`, będący trybem domyślnym), używać najbliższego przesuniętego piksela (tryb filtra `CLAMP`), lub wypełniać przestrzenie kolorem (tryb filtra `COLOR`).

### Krok 1

#### Ustawienie stołu montażowego

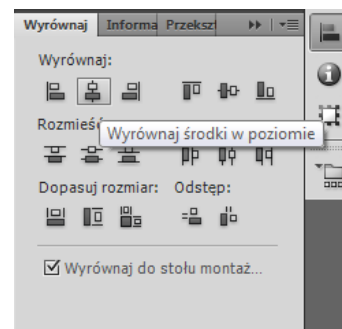
Otwórz nowy plik i ustal rozmiar stołu montażowego na 700X550 pikseli oraz częstotliwość odtwarzania klatek na 30 kl/s.

Wstaw 2 dodatkowe warstwy, z których za chwilę będziesz korzystać. W dolnej warstwie zostanie umieszczone zdjęcie, na warstwie środkowej zostanie umieszczony klip zawierający grafikę będącą mapą kolorów. Górna warstwa przeznaczona jest na ActionScript.

### Krok 2

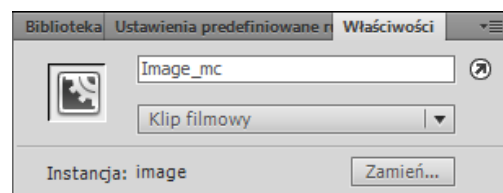
#### Wstawianie zdjęcia

Zaimportuj zdjęcie od razu na stół montażowy na dolną warstwę. Zmień nazwę warstwy na „image”. Dostosuj wymiar zdjęcia tak aby widać było niewielką część stołu i wyśrodkuj je na stole za pomocą panelu **Wyrównaj**.



Konwertuj zdjęcie do klipu (F8).

Nadaj instancji zdjęcia nazwę **Image\_mc**.



### Krok 3

#### Tworzenie skryptu (cz. I) – konstruktor obiektu filtra przemieszczeń

Nazwij górną warstwę „actions”. Otwórz panel **Operacje** i wpisz:

```
1 var dFilter:DisplacementMapFilter = new DisplacementMapFilter ();
```

Polecenie to jest konstruktorem nowego obiektu filtra przemieszczeń o nazwie **dFilter**. Nie określamy od razu w konstruktorze parametrów filtra, gdyż określimy za chwilę wiele cech filtra przez ustawienie jego właściwości.

Jedną z nich jest skala:

```
2 dFilter.scaleX = 50;  
3 dFilter.scaleY = 50;
```

Właściwość ta określa siłę przesunięcia osobno wzdłuż osi x i y. Dla nas jest to liczba określająca powiększenie. Można także użyć wartości ujemnych w celu uzyskania pomniejszenia.

*Uwaga: nie jest to właściwa kolejność określania parametrów filtra tylko wybrana na potrzeby ćwiczenia. Za chwilę pojawią się kolejne linie kodu.*

#### Krok 4

##### Tworzenie skryptu (cz. I) – określenie kanału koloru

Pamiętamy, że właściwość *X component* i *Y component* określają, który kanał koloru obrazu stanowiącego mapę wpływa na położenie x i y pikseli obrazu bazowego. Wartość jest ustanawiana dla trybu RGB i możemy wybrać między `BitmapDataChannel.RED`, `BitmapDataChannel.GREEN` i `BitmapDataChannel.BLUE`. Dla ułatwienia możemy także zastosować wartość 1 (red), 2 (green) lub 4 (blue).

W naszym przykładzie czerwony kanał będzie odpowiadać za przemieszczenie wzdłuż osi x, a zielony kanał za przesunięcie wzdłuż osi Y.

Możemy więc dopisać dwie kolejne linie kodu:

```
4 dFilter.componentX = 1;  
5 dFilter.componentY = 2;
```

#### Krok 5

##### Tworzenie skryptu (cz. I) – wypełnianie przestrzeni kolorem

Teraz określimy jak powinny być wypełniane puste przestrzenie utworzone w wyniku przesunięcia pikseli w inne miejsce. W naszym przykładzie jest to ważne ze względu na możliwość przesunięcia lupy poza zdjęcie.

Mamy do wyboru opcje:

```
DisplacementMapFilterMode.COLOR  
DisplacementMapFilterMode.WRAP  
DisplacementMapFilterMode.CLAMP  
DisplacementMapFilterMode.IGNORE
```

Lub po prostu: "color", "clamp", "wrap", "ignore". Wybierzmy opcję "color", która pozwala kontynuować wyświetlanie pikseli obrazu jeszcze poza jego krawędziami.

```
6 dFilter.mode = "color";
```

#### Krok 6

##### Tworzenie skryptu (cz. I) – wypełnienie i przezroczystość sąsiedztwa obrazu

Ustawimy kolor sąsiedztwa obrazu na wartość `0x000000` i przezroczystość (alpha) na 0. Jest to całkowita przezroczystość, więc nic poza pikselami obrazu nie będzie wyświetlane poza granicami obrazu.

```
7 dFilter.color = 0x000000;  
8 dFilter.alpha = 0;
```

#### Krok 7

##### Tworzenie skryptu (cz. I) – pozycja wyjściowa filtra

Teraz musimy ustawić pozycję wyjściową filtra, czyli w mówiąc w skrócie naszej lupy powiększającej. Robimy to poprzez wprowadzenie obiektu wyznaczającego pozycję x,y filtra.

Rozpoczynamy od tworzenia obiektu o nazwie `dPoint` z parametrami wejściowymi określającymi wartości x i y. Następujący kod dodajemy na początku skryptu:

```
1 var dPoint:Point = new Point(0,0);
```

Punkt ten będzie respektowany, gdy dodamy go do naszego filtra przemieszczeń:

```
11 dFilter.mapPoint = dPoint;
```

```

1  var dPoint:Point = new Point(0, 0);
2  var dFilter:DisplacementMapFilter = new DisplacementMapFilter ();
3
4  dFilter.scaleX = 50;
5  dFilter.scaleY = 50;
6  dFilter.componentX = 1;
7  dFilter.componentY = 2;
8  dFilter.mode = "color";
9  dFilter.color = 0x000000;
10 dFilter.alpha = 0;
11 dFilter.mapPoint = dPoint;
12

```

**Krok 8**

**Tworzenie skryptu (cz. I) – bitmapa kontrolna**

Na koniec pierwszej części pracy ze skryptem utworzymy bitmapę kontrolną i dodamy ją do filtra. Jest to obraz, który zawiera kolorowe piksele adekwatne do określonych wcześniej kanałów koloru odpowiedzialnych za przemieszczenie wzdłuż osi x i y (componentX i Y).

```

12 dFilter.mapBitmap = ;

```

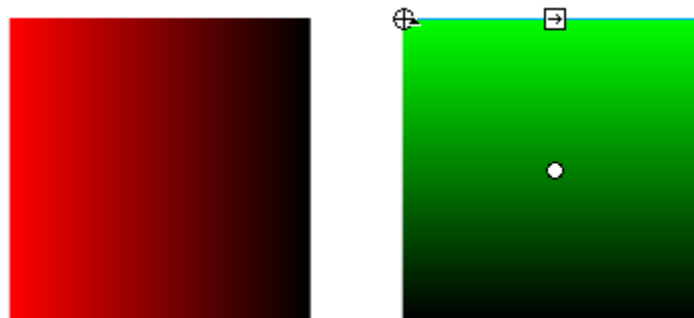
Tą częścią zajmiemy się za chwilę, bowiem musimy utworzyć na stole montażowym obiekt, który będzie kontenerem dla naszej mapy kontrolnej.

**Krok 9**

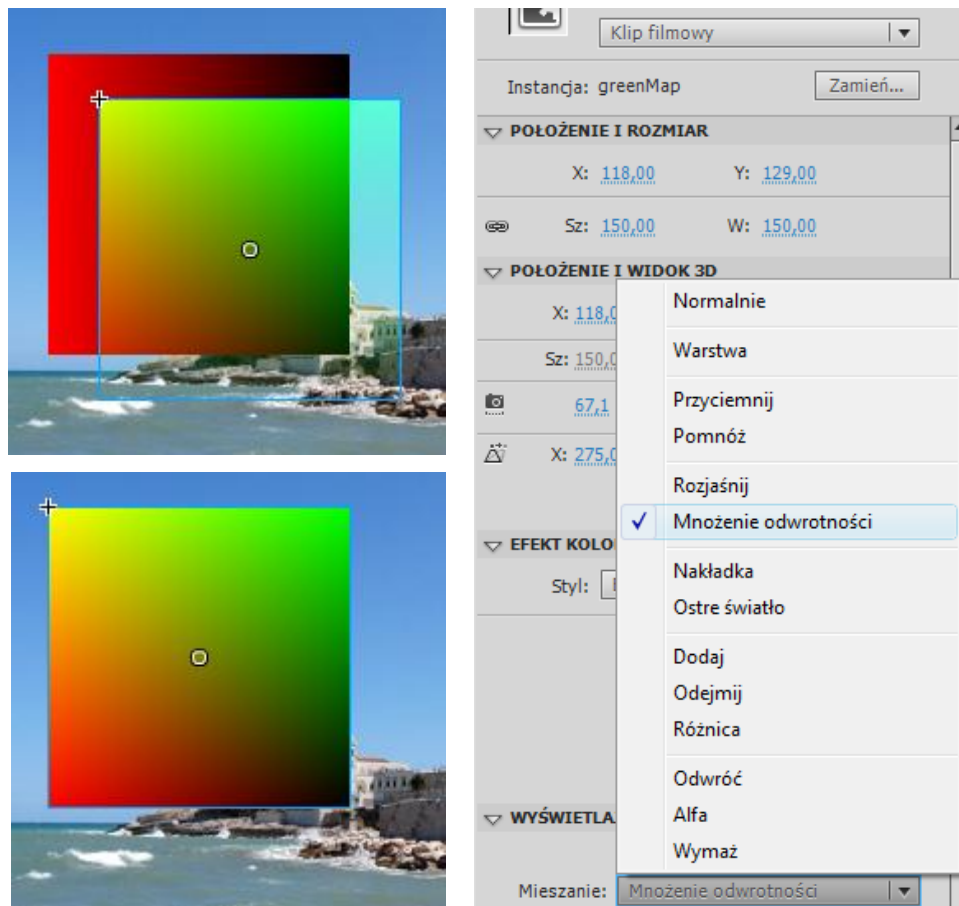
**Tworzenie mapy kontrolnej**

W środkowej warstwie tworzymy kwadrat o boku 150px z wypełnieniem gradientowym liniowym czerwony-czarny. W naszym efekcie to ustawienie kolorystyczne wpłynie na przesunięcie pikseli obrazu w lewo lub w prawo. Konwertujemy kwadrat do klipu filmowego i nadajemy mu nazwę „red map”.

Następnie tworzymy drugi kwadrat tej samej wielkości z wypełnieniem zielony-czarny. W tym przypadku musimy obrócić gradient o 90° (narzędziem przekształcanie swobodne gradientu). To ustawienie z kolei wpłynie na przemieszczenie pikseli wzdłuż osi Y, stąd konieczne jest dokonanie obrotu gradientu. Konwertujemy kwadrat do klipu i nazywamy go „greenMap”.



Mamy teraz dwa oddzielne obrazy kolorów a potrzebujemy tak naprawdę jeden. Dlatego dla klipu „greenMap” ustawiamy wartość **Mieszanie** na **Mnożenie odwrotności**:



W ten sposób każdy kolor klipu „greenMap” zostanie rozjaśniony względem warstwy bazowej, jednak kolor czarny warstwy bazowej jest w tym przypadku „blokada” dla efektu mnożenia. Ułóż klip „greenMap” dokładnie nad klipem „redMap”. Zaznacz oba klipy i przekonwertuj je do jednego klipu o nazwie „colorMap”. Nadaj mu nazwę instancji "colorMap\_mc".

**Krok 10**

**Tworzenie skryptu (cz. II) – Kontener bitmapy kontrolnej**

Wracamy do kodu, aby dokończyć dodawanie bitmapy kontrolnej do filtra dFilter. Na górze skryptu zaraz za konstruktorem punktu dPoint tworzymy nowy obiekt typu BitmapData - nazywamy go "dMap" i ustawiamy wielkość bitmapy według wielkości klipu "colorMap\_mc" (w tym przypadku 150px x 150px). Ustawiamy przezroczystość na "true" i kolor na 0x808080 (szary). Takie ustawienie daje nam pewność, że każdy z pikseli bitmapy będzie neutralny względem mapy kolorów.

```
2 var dMap:BitmapData = new BitmapData(colorMap_mc.width, colorMap_mc.height, true, 0x808080);
```

**Krok 11**

**Tworzenie skryptu (cz. II) – Przechwytywanie mapy kolorów**

Musimy teraz umieścić zawartość klipu colorMap\_mc w obiekcie bitmapData, a także usunąć ten obiekt ze stołu montażowego. Jest to możliwe, gdyż mapa kontrolna została zapisana w obiekcie bitmapData:

```
3 dMap.draw(colorMap_mc);
4 removeChild(colorMap_mc);
```

## Krok 12

### Tworzenie skryptu (cz. II) – Dodanie mapy kontrolnej do filtra

Dadaj bitmapę dMap do obiektu filtra przesunąć dFilter przez ustawienie parametru mapBitmap:

```
dFilter.mapBitmap = dMap;
```

## Krok 13

### Tworzenie skryptu (cz. II) – Dodanie filtra do obrazu

Filtr jest gotowy – mamy już wszystkie ustawienia. Musimy jeszcze dodać go do naszego zdjęcia (Image\_mc). Na końcu kodu ustawiamy parameter filters obrazu jako tablicę:

```
Image_mc.filters = [dFilter];
```

## Krok 14

### Tworzenie skryptu (cz. II) – Ruch szkła powiększającego

Utworzmy teraz efekt podążania lupy za kursorem myszy. Poniżej poprzedniego wiersza kodu wstawiamy:

```
Image_mc.addEventListener(Event.ENTER_FRAME, onFrame)
```

```
function onFrame(e:Event){  
    dPoint.x = mouseX;  
    dPoint.y = mouseY;  
    dFilter.mapPoint = dPoint;  
    Image_mc.filters = [dFilter];  
}
```

Jeśli przetestujemy film, to zobaczymy jak szkło powiększające podąża za myszką. Jednak postaramy się jeszcze zmodyfikować ten efekt dodając niewielką dynamikę. Wymaga to zmodyfikowania dwóch linijek kodu:

```
function onFrame(e:Event){  
    dPoint.x += ((mouseX-colorMap_mc.width/2)-dPoint.x)*0.3;  
    dPoint.y += ((mouseY-colorMap_mc.height/2)-dPoint.y)*0.3;  
    dFilter.mapPoint = dPoint;  
    Image_mc.filters = [dFilter];  
}
```

## Krok 15

### Dodanie ramki do szkła powiększającego

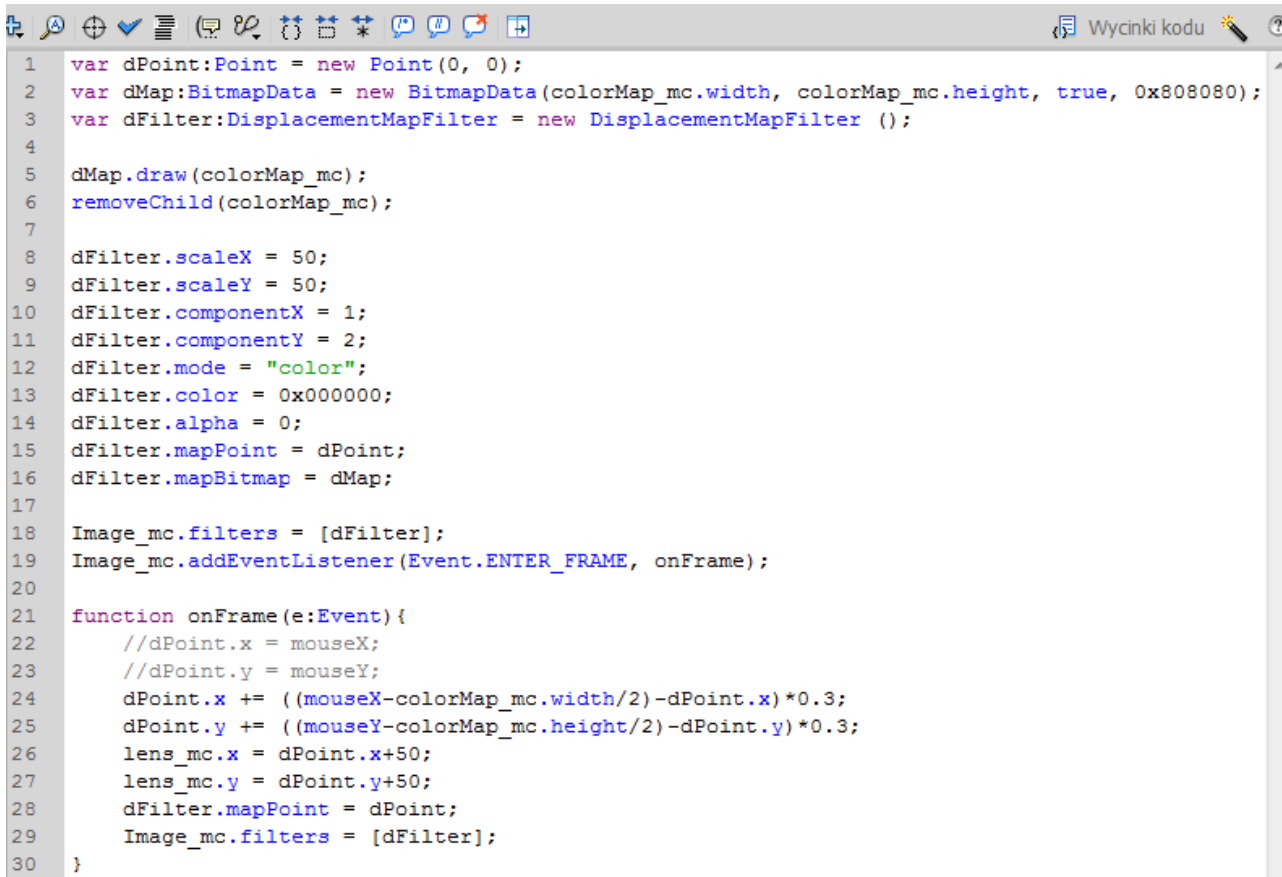
Dodajemy czwartą warstwę poniżej warstwy „action” i nazywamy ją „border”. Rysujemy kwadrat o boku 150px z grubym obramowaniem. Usuwamy wypełnienie kwadratu a obramowanie konwertujemy do klipu o nazwie „ramka”. Klipowi temu nadajemy nazwę instancji „lens\_mc”.



Znów modyfikujemy kod dodając w funkcji onFrame linijki określające położenie klipu „lens\_mc”:

```
function onFrame(e:Event){
    dPoint.x += ((mouseX-colorMap_mc.width/2)-dPoint.x)*0.3
    dPoint.y += ((mouseY-colorMap_mc.height/2)-dPoint.y)*0.3
    lens_mc.x = dPoint.x+50
    lens_mc.y = dPoint.y+50
    dFilter.mapPoint = dPoint
    Image_mc.filters = [dFilter]
}
```

Nasz kod w całości:



```
1 var dPoint:Point = new Point(0, 0);
2 var dMap:BitmapData = new BitmapData(colorMap_mc.width, colorMap_mc.height, true, 0x808080);
3 var dFilter:DisplacementMapFilter = new DisplacementMapFilter ();
4
5 dMap.draw(colorMap_mc);
6 removeChild(colorMap_mc);
7
8 dFilter.scaleX = 50;
9 dFilter.scaleY = 50;
10 dFilter.componentX = 1;
11 dFilter.componentY = 2;
12 dFilter.mode = "color";
13 dFilter.color = 0x000000;
14 dFilter.alpha = 0;
15 dFilter.mapPoint = dPoint;
16 dFilter.mapBitmap = dMap;
17
18 Image_mc.filters = [dFilter];
19 Image_mc.addEventListener(Event.ENTER_FRAME, onFrame);
20
21 function onFrame(e:Event){
22     //dPoint.x = mouseX;
23     //dPoint.y = mouseY;
24     dPoint.x += ((mouseX-colorMap_mc.width/2)-dPoint.x)*0.3;
25     dPoint.y += ((mouseY-colorMap_mc.height/2)-dPoint.y)*0.3;
26     lens_mc.x = dPoint.x+50;
27     lens_mc.y = dPoint.y+50;
28     dFilter.mapPoint = dPoint;
29     Image_mc.filters = [dFilter];
30 }
```

Koniec ćwiczenia

### Dodatkowe objaśnienie techniczne:

Jeśli chodzi o koncepcję i teorię działanie filtra, program Flash Player lub AIR odczytuje pojedynczo piksele na obrazie filtrowanym (tylko w tym obszarze, który podlega filtrowaniu) i wykonuje na każdym pikselu następujące operacje:

1. Odnajduje piksel mu odpowiadający na obrazie stanowiącym mapę. Na przykład obliczenie przez program Flash Player lub AIR wielkości przemieszczenia dla piksela znajdującego się w lewym górnym rogu filtrowanego obrazu odbywa się na podstawie piksela znajdującego się w lewym górnym rogu obrazu stanowiącego mapę.
2. Określa na wartość danego kanału koloru w pikselu na obrazie stanowiącym mapę. Dajmy na to składnik x kanału koloru to kanał czerwieni, tak więc program Flash Player (lub AIR) sprawdza, jaka jest wartość kanału czerwieni na obrazie stanowiącym mapę dla przetwarzanego aktualnie piksela obrazu filtrowanego. Jeżeli obraz stanowiący mapę ma listy kolor czerwony, to kanał czerwieni piksela wynosi 0xFF, inaczej **255. Ta wartość jest używana jako wartość przemieszczenia.**

3. Porównuje ona wartość przemieszczenia do wartości „środkowej” (127, stanowiącej środek między zerem a 255). Jeśli wartość przemieszczenia jest niższa niż wartość środkowa, piksel przesuwa się w kierunku dodatnim (w prawo w przypadku przemieszczenia po x; w dół w przypadku przemieszczenia po y). Z drugiej strony jeśli wartość przemieszczenia jest wyższa niż wartość środkowa piksel przesuwa się w kierunku ujemnym (w lewo w przypadku przemieszczenia po x; w górę w przypadku przemieszczenia po y). **Bardziej precyzyjnie, program Flash Player oraz AIR odejmują wartość przemieszczenia od 127, a wynik (dodatni lub ujemny) jest to wartość względna stosowanego przemieszczenia.**
4. Ostatecznie rzeczywista wartość przemieszczenia jest obliczana na podstawie procentowego udziału, jaki reprezentuje przemieszczenie względne w przemieszczeniu pełnym. Kolor całkowicie czerwony oznacza 100-procentowe przemieszczenie. Jednakże należy wiedzieć, że pełna wielkość przemieszczenia (jeżeli piksel na obrazie mapy jest całkowicie czerwony) powoduje przemieszczenie piksela obrazu filtrowanego na niewielką odległość, zaledwie o tylko pół piksela.
5. **Ten udział procentowy jest następnie mnożony przez wartości skali x lub y w celu wyznaczenia liczby pikseli przemieszczenia, jakie zostanie zastosowane.** Ustawienie skali na 1 spowodowałoby przesunięcie o pół piksela. Ustawiając wartość 250, uzyskujemy odległość przemieszczenia wynoszącą około 125 pikseli.

Można to też przedstawić za pomocą wzoru:

$$\text{dstPixel}[x,y] = \text{srcPixel}[x + ((\text{componentX}(x,y) - 128) * \text{scaleX}) / 256, y + ((\text{componentY}(x,y) - 128) * \text{scaleY}) / 256)]$$

gdzie `componentX(x,y)` pobiera wartość składowej koloru określonej we właściwości `componentX` dla danego piksela mapy `mapBitmap` w `(x - mapPoint.x , y - mapPoint.y)`.

Inaczej mówiąc przesunięcie piksela pod względem jednej składowej odbywa się według następującego przeliczenia:

$$\text{Przesunięcie piksela} = (\text{Składowa} - 128 / 256) * \text{skala}$$

Składowa to R, G lub B określona dla `componentX` (przesunięcie w poziomie) lub `komponentY` (przesunięcie w pionie). Skala to analogicznie `scaleX` lub `scaleY`.

Poniższa tabela pokazuje jakie rzeczywiste przesunięcie piksela i w jakim kierunku uzyskamy dla składowej o określonej wartości, przy różnych wariantach skali. Rozpatrujemy `componentX`.

R	skala	Przesunięcie (px)	Kierunek Dla componentX
255	1	0,5	lewo
255	50	25	lewo
255	100	50	lewo
255	150	75	lewo
255	250	125	lewo
190	1	0(0,2)	lewo
190	50	12	lewo
190	100	24	lewo
190	150	36	lewo
190	250	60	lewo
127	1	0	brak
127	50	-0,2	prawo
127	100	-0,4	prawo
127	150	-0,6	prawo
127	250	-1	prawo
0	1	-0,5	prawo
0	50	-25	prawo
0	100	-50	prawo
0	150	-75	prawo
0	250	-125	prawo