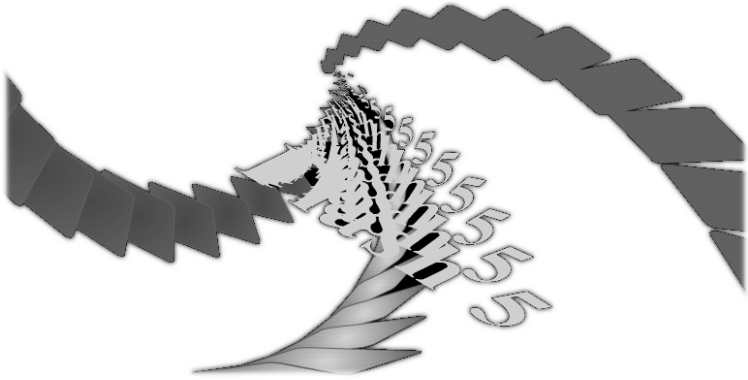
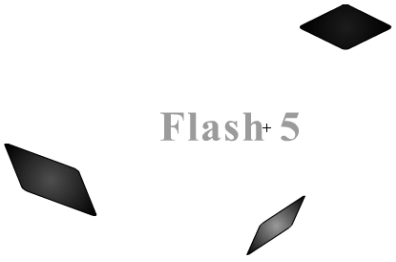
5.2 Effekte

Ordner Texteffe, Datei: texteffe.fla

Timeslide (Texteffekt)

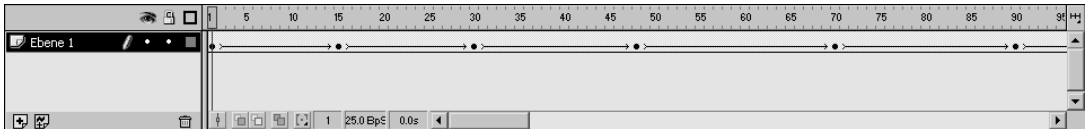


Anhand des folgenden Beispiels zeigen wir Ihnen, wie Sie mit einfachen Mitteln einen eindrucksvollen und doch wenig speicherintensiven Texteffekt erzeugen können. Hierfür werden der Tweeningeffekt, verschiedene Farbeinstellungen und der `duplicateMovieClip`-Befehl benutzt.

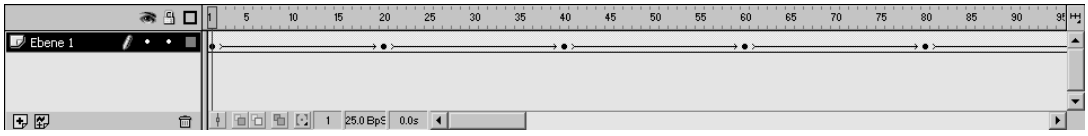


Der ganze Effekt besteht aus einer Grafik. Diese wird mit Hilfe des Tweeningeffektes bewegt. Damit dies nicht direkt auf der Hauptzeitleiste passiert und weil die Grafik dupliziert werden muss, baut man diese mit dem Bewegungstweening in ein Movie ein. Das Movie hat den Instanznamen *Grafik*.

In diesem Movie befindet sich ein weiteres Movie, da auf die Grafik noch ein Farbeffekt angewendet wird.



Im zweiten Movie befindet sich ein weiteres Tweening, bei dem die Farbwerte der Grafik verändert werden.



Man erkennt sofort, dass die Punkte des Farbwechsels und des Richtungswechsels asynchron verlaufen; die zwei Movies haben auch unterschiedliche Längen. Dadurch erhält man den Effekt, dass bei gleichartigen Bewegungen nicht immer die selben Farbverläufe entstehen.

Sie können dem Ganzen nun Tiefe bzw. einen dreidimensionalen Charakter verleihen, indem Sie das Movie *Grafik* mit dem `duplicateMovieClip`-Befehl mehrmals duplizieren und die duplizierten Movies ein wenig umpositionieren, drehen, verkleinern oder vergrößern.

Mit diesem Effekt lassen sich mit geringen Mitteln und kleinen Dateigrößen beeindruckende Movies erzeugen. Man kann ihn auch beim Preloader einsetzen, um den Ladestatus anzuzeigen, indem man keinen statischen Text sondern einen dynamischen Text benutzt, der z. B. immer die aktuelle Prozentzahl angibt.

Das ActionScript mit dem `duplicateMovieClip`-Befehl, das im ersten Frame auf der Hauptbühne liegt, sieht wie folgt aus:

```
for (i=1; i<15; i++) {
    duplicateMovieClip ("_root.grafik", "grafik"+i, i);
    _root["grafik"+i]._yscale   =i*i;
    _root["grafik"+i]._xscale   =i*12;
    _root["grafik"+i]._rotation =i*8;
    _root["grafik"+i]._y        =i*10+150;
}
_root.grafik.gotoAndStop(1);
stop ();
```

Hier werden 14 Kopien des Movies *Grafik* angefertigt und deren Position variabel angepasst, je nachdem, um die wievielte Kopie es sich handelt. Die späteren Kopien werden auf dem Bildschirm immer größer dargestellt, da i sich erhöht und dadurch die errechneten Werte für den `_xscale` oder `_yscale` auch vergrößern. Leider beansprucht die Darstellung des Bewegungsablaufs mit mehreren duplizierten Movies sehr viel Rechnerleistung, so dass man rasch an die Grenze des Machbaren gelangt und das SWF beim

Abspielen unter Umständen ruckelt. In der vorletzten Zeile wird das SWF, das außerhalb der Bühne liegt gestoppt, damit dieses nicht aus Versehen im Hauptbild auftaucht.

Texteffekt.fla

▶ **Texteinblendung**

Um eine schlichte Texteinblendung zu verwirklichen, verwenden einen Maskeneffekt. Legen Sie zunächst eine Filmsequenz an, in der Ihr Text in einem statischen Textfeld als Maske angelegt ist.

In der maskierte Ebene erstellen Sie ein Bewegungstweening mit einem Verlauf (Hintergrundfarbe zu Textfarbe) mit anschließender Fläche in der gewünschten Textfarbe.

Das Ergebnis sieht wie folgt aus:

Wenn Sie diese nun abspielen, sieht der Effekt wie folgt aus:

Mauseffekte/Maustrailer

Maustrailer sind Effekte, die in Abhängigkeit zum Mauszeiger stehen. Dies kann ein Fadenkreuz, ein rotierender Stern oder auch die Ausgabe von Mauskoordinaten sein.

▶ **Mauskoordinaten direkt am Mauszeiger ausgeben (Flash 4)**

Dieser Effekt veranschaulicht das Zuordnen von Variablen sowie das Zusammensetzen von String-Ausdrücken.

Zuerst erstellen Sie auf der Hauptbühne einen Movieclip mit dem Instanznamen „trail“. Dieser Name ist beliebig, wird aber im Folgenden verwendet. Der erste Frame auf der Hauptbühne enthält folgendes ActionScript:

```
Start Drag ("/trail", lockcenter)
```

Hier wird der soeben erstellte Movieclip an die aktuellen Mauskoordinaten geheftet. Der Movieclip „trail“ selbst enthält ein Textfeld, das den Variablennamen „ausgabe“ erhält. Im ersten Frame des Movieclips werden die Mauskoordinaten mit Hilfe des Movieclips ausgelesen, in einen String gesetzt und dem Textfeld zugeordnet:

```
Set Variable: "X" = GetProperty ("",_x)
Set Variable: "Y" = GetProperty ("",_y)
Set Variable: "ausgabe" = X&", "&Y
```

Mit Hilfe von & werden Variablen und Textausdrücke verkettet. Damit die Koordinatenangaben ständig aktualisiert werden, muss im darauf folgenden Frame die folgende Aktion eingetragen werden:

```
Go to and Play (1)
```

Dieser Befehl bewirkt, dass eine Endlos-Schleife entsteht und somit die Werte laufend aktualisiert werden.

▶ Alternative Mauszeiger erzeugen (Flash 5)

Flash 5 verfügt über die Möglichkeit, den System-Mauszeiger über dem aktivem Flashfilm auszublenden. Der entsprechende ActionScript-Befehl lautet:

```
Mouse.hide();
```

Um dem Benutzer weiterhin die Möglichkeit der Navigation zu bieten, muss an Stelle des Mauszeigers ein anderes Symbol eingesetzt werden. (Ausnahme sind z. B. Spiele mit Tastatursteuerung.)

Mit Hilfe des `onClipEvent (Ereignis)` frägt der Film, der den Mauszeiger ersetzen soll, automatisch laufend ab, ob

- der Cursor bewegt wurde (`mouseMove`) – wenn ja, wird der alternative Mauszeiger nachgeführt.
- eine Maustaste gedrückt (`mouseDown`) wird – wenn ja, nimmt der Cursor das Erscheinungsbild an, das im Frame mit dem Namen „pressed“ hinterlegt wurde.
- eine Maustaste losgelassen (`mouseUp`) wird – wenn ja, nimmt der Cursor das Erscheinungsbild an, das im Frame mit dem Namen „released“ hinterlegt wurde.

Beim Laden des Cursors (`load`) bleibt der System-Cursor verborgen. Die Anweisung `stop()` bewirkt, dass nicht alle möglichen Cursor, die in der gleichen Filminstanz liegen, durchlaufen werden.

```
onClipEvent (load) {
    Mouse.hide();
    stop();
}
```

Sobald die Maus bewegt wird (`mouseMove`), werden die X-Y-Koordinaten auf die Instanz angewandt. Mit `_x` werden die instanzeigenen Koordinaten mit denen des Mauscursors (unsichtbar) gleichgesetzt (`_root._xmouse`). Da wir davon ausgehen, dass sich der Mauszeiger nachher an der Hauptbühne ausrichten soll, legt `_root` den Bezugspunkt fest, der auch für die Bestimmung von`_xmouse` herangezogen werden muss. Gleiches gilt für die Y-Koordinate.

```
onClipEvent (mouseMove) {
    _x = _root._xmouse;
    _y = _root._ymouse;
    updateAfterEvent();
}
```

Ist eine Maustaste gedrückt, springt die Instanz in den Frame „pressed“, wo ein entsprechendes Bild eingefügt ist. Dort bleibt der Film stehen, so lange nicht einer der anderen Events (`mouseUp`) eintritt.

```
onClipEvent (mouseDown) {
    this.gotoAndStop("pressed")
    updateAfterEvent();
}
```

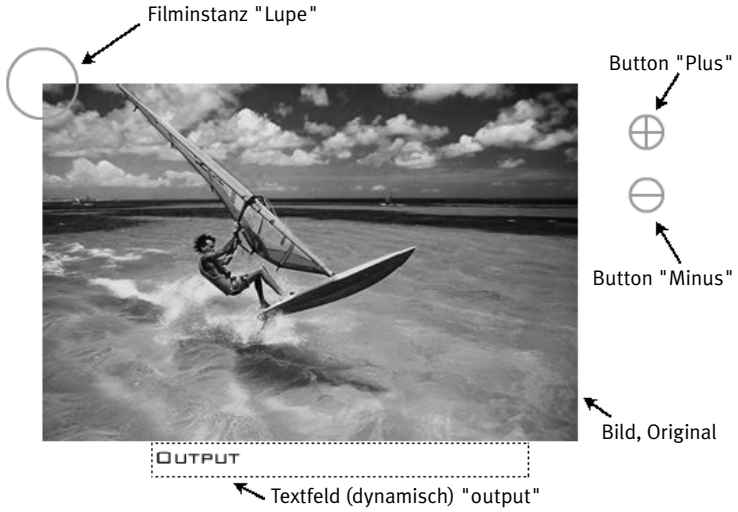
Wird die Maustaste losgelassen, springt die Instanz in den Frame „released“, wo ein entsprechendes Bild (normalerweise das Erscheinungsbild des „Ausgangscursors“) eingefügt ist. Dort bleibt der Film stehen, so lange nicht einer der anderen Events (`mouseDown`) eintritt.

```
onClipEvent (mouseUp) {
    this.gotoAndStop("released");
    updateAfterEvent();
}
```

Maskeneffekte (Lupe)

Lupe.fla

Eine Frage, die immer wieder gestellt wird, ist, wie mit Flash eine Lupe zu realisieren ist. Auf den ersten Blick ist dies wenig problematisch, doch sehr bald stellen Sie fest, dass es nicht möglich ist, per `startDrag` eine Maske über den maskierten Bereich zu ziehen. An dieser Stelle kann man aber zu einem Trick greifen: Legen Sie eine Instanz für die Lupe an und kopieren Sie hier Ihr Originalbild hinein. Dieses ist unter der Maske per ActionScript ansprechbar und „beweglich“. Wenn Sie nun Original und Vergrößerung aufeinander abstimmen, erhalten Sie Ihren Lupeneffekt. Die folgende Abbildung zeigt die Anordnung aller Objekte auf der Hauptbühne:



Hinter den Buttons verbergen sich nur kurze Scripts, um die Variable *faktor* zu verändern.

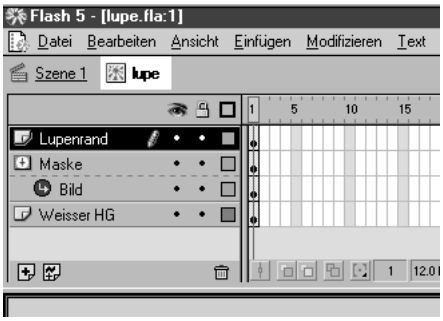
Button Minus:

```
on (press){
_level0.lupe.faktor+=10;
}
```

Button Plus:

```
on (press){
_level0.lupe.faktor-=10;
}
```

Die aktuellen `faktor`-Werte werden später immer in der `output`-Anzeige angegeben. Die Lupe selbst enthält vier Ebenen, die wir im Folgenden erläutern:



Die oberste Ebene *Lupenrand* beinhaltet ausschließlich den Rahmen der Lupe, oder vielmehr die Fassung. Die zweite Ebene *Maske* enthält eine Maske,die genau passend zum Rand sein sollte. Die dritte Ebene enthält die Instanz *Bild* und das Bild surfer.jpg in Originalgröße; die vierte und unterste Ebene enthält eine Kopie der Maske in der Farbe des Hintergrundes, um am Rand des Bildes dieses nicht doppelt zu sehen. Folgendes ActionScript steuert nun die Aktionen:

```
onClipEvent (load) {
     startDrag ("", true);
     var faktor = 200;
```

Die Lupe wird an den Cursor geheftet, der Zoom auf 200% voreingestellt.

```
}
onClipEvent (enterFrame) {
     _level0.lupe.bild._x = (_level0.lupe.faktor/
100)*_level0.ursprung._x-(_level0.lupe.faktor/
100)*(_root._xmouse);
     _level0.lupe.bild._y = (_level0.lupe.faktor/
100)*_level0.ursprung._y-(_level0.lupe.faktor/
100)*(_root._ymouse);
     _level0.lupe.bild._xscale = (_level0.lupe.faktor);
     _level0.lupe.bild._yscale = (_level0.lupe.faktor);
```

Die Koordinaten der Bilder (_levelo.ursprung und _levelo.lupe.bild) werden abgestimmt, die Vergrößerung gesetzt.

```
     _level0.output = _level0.lupe.faktor+"% des Orginals";
```

Der aktuelle Zoomfaktor wird ausgegeben.

```
}
```

Dieses Script liegt in der Instanz *Lupe* auf der Hauptbühne.

Soundeffekte

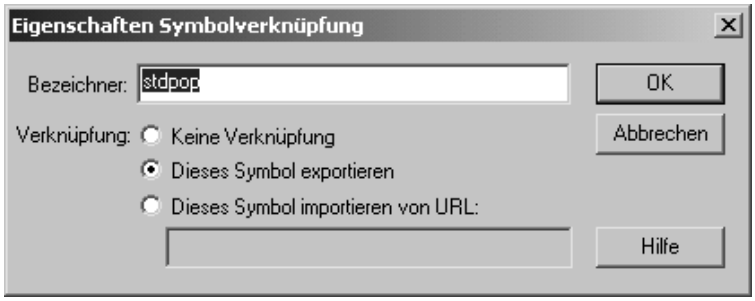
Ordner Soundeff, Dateien: soundeff.fla und flugzeug.fla

In Flash 5 ist es möglich, mit ActionScript die Volume(Lautstärke) und Pan(Balance)-Einstellungen zu ändern. Dies geschieht mit einem Soundobjekt.

Nachdem die wichtigsten Grundkenntnisse im Abschnitt *Sound* → 62 schon erläutert wurden, kommen wir nun direkt auf das Wesentliche des Soundobjektes. Die erste Frage, die Sie sich vielleicht stellen könnten, wäre, wozu das Soundobjekt überhaupt benötigt wird. Die zwei häufigsten Anwendungen sind einerseits die Erstellung eines Pan(Balance)-Reglers bzw. eines Volume(Lautstärke)-Reglers, andererseits die Unterlegung von User-gesteuerten Grafiken mit Soundeffekten, bei denen sich Balance und Lautstärke anpassen.

Um das Soundobjekt überhaupt benutzen zu können, müssen Sie das entsprechende Sample vorab diesem Soundobjekt zuweisen. Hierfür müssen Sie dem Sample einen Verknüpfungsnamen geben.

- Importieren Sie ein Sample in Flash (→ 62 *Sound*).
- Öffnen Sie die Bibliothek Ihres Filmes (STRG + L).
- Klicken Sie mit der rechten Maustaste auf das Sample in der Bibliothek Ihres Filmes und wählen Sie anschließend die Option VERKNÜPFUNG .



- Wählen den Menüpunkt VERKNÜPFUNG › DIESES SYMBOL EXPORTIEREN aus.
- Bei Bezeichner geben Sie den Namen, den die Verknüpfung haben soll, ein.
- Bestätigen Sie mit „OK“.

In diesem Beispiel wurde dadurch dem Sample der Verknüpfungsname `stdpop` zu gewiesen.

Um den Sound zu starten, erstellen Sie zunächst ein Soundobjekt. Dieses Soundobjekt wird wie alle anderen Objekte auch mit `new name` erstellt. Da das Objekt am besten direkt zu Beginn hinzugefügt wird, macht man dies am besten über `onClipevent(load)`. Da Clipevents aber leider nur auf Movies angewendet werden können, sollte man sich ein Movie in dem Film aussuchen, in dem man alle Clipevents einbaut, so dass man die Befehle auch wieder findet. Es spricht allerdings nichts dagegen, einen Befehl in ein anderes Movie einzubauen. Allerdings sollten alle `onClipevent(load)`, in denen eventuel noch Funktionen definiert werden, zusammen in einem Movieclip liegen.

On ClipEvent
→*Kapitel 2: Neuerungen in Flash 5.*

```
onClipEvent (load) {
    standardpop = new Sound();
    standardpop.attachSound("stdpop");
}
```

`onClipevent (load)` besagt, dass der folgende Befehl einmal ausgeführt wird – und zwar dann, wenn der Movieclip direkt zu Beginn eingeladen wird.

Der erste Befehl erzeugt ein Soundobjekt mit dem Namen „standardpop“. Der zweite Befehl übergibt das Sample mit dem Verknüpfungsnamen `stdpop` an das Soundobjekt. Um nun das Musiksample zu starten oder zu stoppen, sprechen Sie direkt das Soundobjekt an.

```
on (release) {
standardpop.stop()
}
```

Stoppen oder Starten des Sounds:

```
on (release) {
    standardpop.stop()
    standardpop.start(0,20);
}
```

Der Stoppbefehl steht dort nur, damit das Sample nicht doppelt abgespielt wird, wenn der Besucher zweimal auf Start drückt. Das Sample wird stattdessen gestoppt und anschließend neu gestartet. Die Parameter, die mitübergeben werden, sind die Startposition und die Anzahl der Wiederholungen. Dieses Sample würde direkt am Anfang des Samples starten und 20mal wiederholt werden.

Das Soundobjekt bietet aber noch eine Vielzahl weiterer Möglichkeiten: Zum Beispiel kann man die Eigenschaften Lautstärke (Volumen) und Balance (Pan) auslesen. Dies geschieht auch in der Beispielsdatei bei den beiden Textfeldern. Da man für die Textfelder wieder ein `onClipEvent` braucht, erstellt man beide Textfelder direkt in einem Movieclip. Theoretisch hätte man diese Befehle auch in den ersten Movieclip einbauen können, man hätte dann aber absolute Pfadangaben machen müssen. Außerdem wird die Gliederung auf diese Weise übersichtlicher. Dies ist aber natürlich nicht zwingend notwendig. Wir haben die Befehle in einem eigenen Movieclip erstellt, um die Übersicht zu behalten. Diese Vorgehensweise ist aber nicht zwingend. Durch das Erstellen eines einzelnen Movies nur für die Textboxen wird dies aber sofort ersichtlich.

```
onClipEvent (enterFrame) {
    vol = _root.standardpop.getVolume();
    pan = _root.standardpop.getPan();
}
```

`OnClipEvent (enterFrame);` Diese Aktion sorgt dafür, dass die enthaltenen Befehle immer wieder abgearbeitet werden. In diesem Fall werden den Variablen `vol` und `pan` immer wieder die Ergebnisse der Funktionen `getPan` und `getVolume` zugewiesen. GetPan gibt die Einstellungen der Balance wieder. Dieser Wert kann zwischen -100 und 100 liegen und wird als ganze Zahl ausgegeben. `GetVolume` gibt die Einstellung der Lautstärke wieder. Dieser Wert liegt zwischen 0 und 100 und wird als ganze Zahl ausgegeben. Da die Textfelder die Zuweisung zu den Variablen `vol` und `pan` haben, zeigen sie immer die aktuellen Einstellungen der Lautstärke und der Balance an.

Den interessanteren Teil des Soundobjekts dürfte der Zugriff auf die Einstellungen des Sounds über das ActionScript darstellen. In der Beispielsdatei haben wir zwei Regler erstellt, über die man die Lautstärke wie auch die Balance ändern kann.

Wir erklären zunächst das Funktionsprinzip eines solchen Reglers. Der erstellte Regler ist vom ActionScript her sehr universell einsetzbar, da man das ActionScript nicht ändern braucht, um die Länge zu ändern. Zuerst er-

kennt man wieder ein Movieclip, um das Ganze zusammenzuhalten, damit die einzelnen Objekte nicht direkt auf der Hauptbühne liegen. Theoretisch könnte man dieses Movieclip auch als Smartmovieclip der Bibliothek hinzufügen. In diesem Movie befinden sich zwei weitere Movies: eines mit dem Instanznamen *Länge*, das die Strecke des Reglers enthält, zum anderen der Regler selbst, der den Instanznamen *Regler2* besitzt. Will man einen Regler mit einem längeren Weg erzeugen, muss man so beim Erstellen lediglich mit der Maus die Länge des Movies verändern. Würde es sich nicht um ein Movieclip handeln, wäre dies so nicht möglich. Der Regler liegt in einem Movie, damit dieser mit der Aktion startDrag an den Mauszeiger „geheftet" werden kann.

Der Regler ist als Button ausgeführt und besitzt folgendes ActionScript:

```
on (press) {
    startDrag (this, false, _parent.laenge._x -
_parent.laenge._width/2, _parent.laenge._y, _parent.laenge._x +
_parent.laenge._width/2, _parent.laenge._y);
}
on (release, releaseOutside) {
    stopDrag ();
}
```

Beim Drücken des Schalters wird also `startDrag` ausgeführt.

Die Aktion `startDrag` besitzt folgende Argumente:

Allgemein ausformuliert: `StartDrag` (Instanzname,Objekt unter Mauszeiger zentrieren, true/false, max. linker X-Wert für das Objekt, max. oberer Y-Wert für das Objekt, max. linker X-Wert für das Objekt, max. unterer Y-Wert für das Objekt);

Die `startDrag`-Aktion wird auf das Movie „this" angewandt, `This` ist ein Synonym für, das Movie, in dem sich der Befehlsaufruf befindet. Der kleinste X-Wert entspricht dem `_parent.laenge._x - _parent.laenge._width/2`, also der X-Position des Movieclips *Länge* weniger der halben Breite des Movieclips *Länge*. Dadurch erhält man den linken äußeren Punkt, bis zu dem der Regler maximal gezogen werden darf. Der rechte Rand errechnet sich genau auf die gleiche Weise, nur dass man die Hälfte nicht abzieht sondern dazu addiert. Da der Regler nur waagerecht verschoben werden soll, wird die Y-Position für den Regler auf einen konstanten Wert gesetzt. Damit der Regler genau mittig erscheint, wird dieser auf den Y-Wert des Movieclips *Länge* gesetzt. Wenn man nun den Movieclip *Länge* verändert, verändert man auch die Reichweite des Reglers.

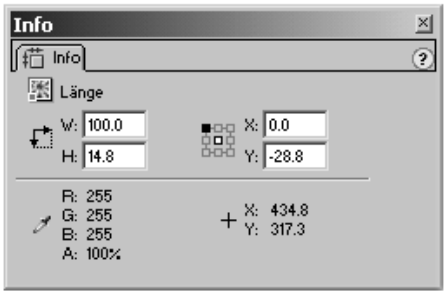
Anstatt den Rand des Movies über die Position und die Breite zu ermitteln, kann man auch das getBounds-Objekt verwenden.

Die Aktion `stopDrag` wird beim Loslassen der Maustaste ausgeführt und bewirkt, dass der Movieclip *Regler* nicht mehr dem Mausezeiger folgt.

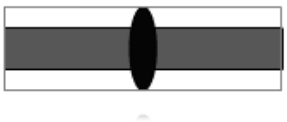
Dadurch hat man nun einen beweglichen Regler. Dieser hat aber noch keine Funktion. Um die Lautstärke analog zu der Reglerstellung zu ändern, muss man fortlaufend die Stellung des Reglers überprüfen und die Stellung der Lautstärke zuweisen.

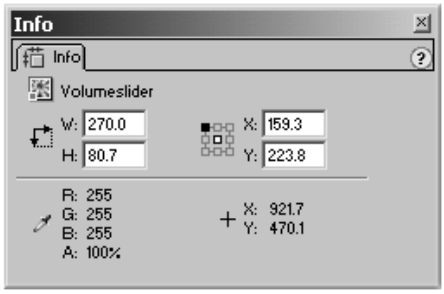
```
onClipEvent (enterFrame) {
    _root.standardpop.setVolume(Regler2._x);
}
```

Durch diesen Befehl entspricht die Lautstärke des Sounds *standardpop* (Verknüpfungsname) stets der Reglerstellung. Da die Lautstärke zwischen 0 und 100 liegen sollte, muss man nun noch dafür sorgen, dass die Minimalstellung des Reglers x = 0 beträgt und die Maximalstellung x=100. Wenn man die Breite bzw. die Länge des Movieclips *Länge* genau auf 100 setzt und die x-Position auf 0, entspricht dies genau den gewünschten Parametern. Ändert man nun im übrigen auf der Hauptbühne die Größe des Movies *Volume*, hat dies keinen Einfluss auf die Größe der darin liegenden Movies. Da der Movieclip *Länge* in dem Movieclip *Volume* sitzt, der auf der Hauptbühne verbreitert wurde, scheint es so, als ob die Reglerbreite sich ebenfalls vergrößert hätte. Dies ist aber nur optisch der Fall, die X-Y Koordinaten in den Movieclips bleiben erhalten, so dass der Movieclip *Länge* weiterhin eine Breite von 100 besitzt.



In dem Movieclip `Regler` besitzt der Movieclip L*änge* genau eine Breite von 100 und genau auf dieser Grundlage wird die X-Position des Reglers abgefragt.

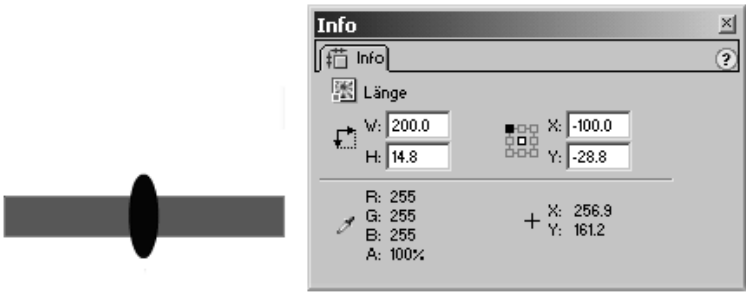




Auf der Hauptbühne wurde der Movie-Regler anschließend gestreckt, damit der Regler groß genug auf dem Bildschirm erscheint. Dies hat aber keinen Einfluss auf die darin liegende Abfrage und die X-Werte.

Für den Balance- bzw. Panregler gelten fast genau dieselben Einstellungen. Einzig und alleine der Movieclip *Länge* in dem Movieclip *Regler* wird

abgeändert, da der Panregler nicht Werte von 0 bis 100 erzeugen soll, sondern von -100 bis 100. Wichtig dabei ist die X-Position und die Breite. Die Y-Position sowie die Höhe sind irrelevant, da dem Regler ohnehin eine feste Y-Position zugewiesen wird. Deshalb sehen die Einstellungen wie folgt aus:



Nun, da die Lautstärke und die Balance jeweils automatisch an die Position des Reglers angepasst werden, kann man diese Werte per ActionScript noch automatisch verändern lassen. Dies ist sehr nützlich, wenn man auf einer Homepage den Sound bzw. die Musik ein- oder ausblenden möchte. Da wir in diesem Beispiel Regler benutzt haben, setzen wir immer direkt die X-Werte der Regler ein, um die Lautstärke- oder Paneinstellungen zu ändern. Wenn Sie die Regler nicht benutzen, müssen Sie die entsprechende Zeile im ActionScript, in dem die X-Position des Reglers angegeben wird, gegen eine entsprechende Soundvolumenangabe tauschen.

Der Schalter zum Einblenden eines Samples befindet sich in einem Movieclip, welches direkt im ersten Frame stoppt. Wird der Schalter gedrückt, so springt der Movieclip in den zweiten Frame. In diesem wird dann folgendes ActionScript ausgeführt.

```
vol=_root.Reglervol.Regler2._x;
_root.fadeout.gotoAndStop ( 1 );
_root.special.gotoAndStop ( 1 );
```

Zuerst wird in der Variable `vol` die X-Position des Lautstärkereglers gespeichert. Diese Position entspricht der augenblicklichen Lautstärke.

Die zwei folgenden `gotoAndStop (1);`-Anweisungen sorgen dafür, dass diese Aktion abgebrochen wird, falls zuvor die Ausblende- oder die Special-Taste gedrückt wurde.

Im dritten Frame wird zuerst geprüft, ob die maximale Lautstärke von 100 bereits erreicht ist. Falls dem nicht so ist, wird die Lautstärke um 1 erhöht und zum nächsten Frame gesprungen, welcher wieder auf den zweiten Frame verweist. Das Ganze wiederholt sich solange, bis die maximale Lautstärke von 100 erreicht wurde; dann wird der Abspielvorgang des Einblende-Movies gestoppt.

```
if (vol>=100) {
    gotoAndStop (1);
} else {
    vol = _root.Reglervol.Regler2._x;
    vol++;
    _root.Reglervol.Regler2._x = vol;
}
```

Die Befehlszeile `vol = _root.Reglervol.Regler2._x;` steht dort, damit der User auch während des Ausblendvorgangs die Möglichkeit hat, mit der Maus die Reglerposition zu ändern.

Ohne diese Zeile würde der Regler kontinuierlich auf 100 zusteuern. So wird zwischendurch noch einmal abgefragt, ob der Regler seine Position geändert hat und falls dies zutrifft, wird die Variable `vol` diesem neuen Wert angepasst.

Nebenbei bemerkt kann man all diese Befehle genauso gut in einem `onClipEvent()` erstellen.

Der Button für das Ausblenden ist nach genau demselbem Schema aufgebaut. Der einzige Unterschied: Die Variable `vol` wird nicht aufsteigend, sondern absteigend gezählt. Entsprechend wird das Abspielen des Movies bei 0 angehalten und nicht bei 100.

```
if (vol<=0) {
    gotoAndStop (1);
} else {
    vol = _root.Reglervol.Regler2._x;
    vol--;
    _root.Reglervol.Regler2._x = vol;
}
```

Der Special-Schalter basiert ebenfalls auf dem gleichen Prinzip, da sich fast alles weitere mit diesem Aufbau erzeugen lässt. Es wurden nur zwei weitere Variablen eingeführt, um die Laufrichtung der Regler zu bestimmen – `VolUp` und `PanUp`.

```
if (vol<20) {
volup=true;
} else if (vol>100) {
volup=false;
}
if (volup==true) {
vol++;
} else {
vol--;
}
_root.slidervol.slider2._x=vol;
```

```
if (pan<-80) {
panup=true;
} else if (pan>80) {
panup=false;
}
if (panup==true) {
pan++;
} else {
pan--;
}
_root.sliderpan.slider2._x=pan;
```

Da in diesem Frame nicht die Position der Regler neu abgefragt wird, ist ein Eingreifen des Users während des Ablaufens des Effektes nicht möglich.

Auf der CD befindet sich auch noch eine zweite Datei mit einem Beispiel für das Soundobjekt.

Das Soundobjekt besitzt noch die Methode `setTransform`. Mit dieser Methode ist es möglich, auf die Ausgabekanäle des Sounds zuzugreifen: z.B. besitzt ein Stereo-Sample zwei Kanäle (rechts und links). Normalerweise wird das Sample so abgespielt, dass der rechte Kanal des Samples auf dem rechten Ausgang liegt und der linke Kanal auf dem linken Ausgang.

Der Aufruf des `SoundTransform`-Objektes sieht wie folgt aus.

```
on (release) {
TransformObjektname = new Objekt();
TransformObjektname= { ll: '100', lr: '0', rr: '100', rl: '0'};
Sound.setTransform(TransformObjektname);
Sound.start(0,1);
}
```

Erstellen Sie zuerst ein Objekt. Dies geschieht in der ersten Zeile. Das Objekt hat hier den Namen *TransformObjektname*. In der zweiten Zeile werden dem Objekt Werte zugewiesen.

`LL`	steht für den linken Kanal des Samples; das zweite L gibt an, dass dieser auf dem linken Ausgang wiedergegeben wird.
`LL`	bedeutet linker Kanal auf dem linken Ausgang (Standard = 100).
`LR`	bedeutet linker Kanal auf dem rechten Ausgang (Standard = 0).
`RR`	bedeutet rechter Kanal auf dem rechten Ausgang (Standard = 100).
`RL`	bedeutet rechter Kanal auf dem linken Ausgang (Standard = 0).

Die Zuordnung funktioniert prozentweise. Man sollte darauf achten, einem Ausgang nie über 100 zuzuweisen, da es sonst zu einer leichten Übersteuerung kommt.

Im Normalfall wird also der rechte Kanal dem rechten Ausgang zu geordnet und der linke Kanal dem linken Ausgang. Diese Einstellungen kann man nun verändern. Dadurch kann man bei einem Stereosampel den Stereoeffekt umkehren. Dafür würden die Einstellungen wie folgt aussehen:

```
TransformObjektname= { ll: '0', lr: '100', rr: '0', rl: '100'};
```

Bei diesem Befehl werden die Kanäle gewechselt. Möglich wäre auch den Stereoeffekt „auszuschalten“:

```
TransformObjektname= { ll: '50', lr: '50', rr: '50', rl: '50'};
```

Man kann aus einem Stereosound auch einen Monosound machen.

```
TransformObjektname= { ll: '100', lr: '100', rr: '0', rl: '0'};
```

Hierbei würde jetzt der linke Kanal auf beiden Ausgängen wieder gegeben, der rechte Kanal wird nicht ausgegeben.

Erst durch die Zuordnung des Objektes *TransformObjektname* zu der Methode `setTransform` und dem davor zu setzenden Soundobjekt wird dieser Befehl aktiv.

```
Sound.setTransform(TransformObjektname);
```

Dieses Objekt funktioniert also wiederum nur mit dem Soundobjekt zusammen. Die Methoden `setPan` und `setVolume` zu `setTransform` überschreiben sich gegenseitig.

Hat man z.B. die Lautstärke auf 70 und die Balance auf 50 gesetzt und wendet nun `setTransform` an, so werden die Werte von `setPan` und `setVolume` ungültig und es gelten die neuen Einstellungen von `setTransform`. Dies gilt natürlich auch umgekehrt.

5.3 Erweiterte Funktionen

Externe Dateien einlesen

Es ist möglich, in Flash „externe“ Daten zu importieren. Diese Daten müssen allerdings nach einer bestimmten Syntax aufgebaut sein. Des Weiteren erläutern wir das Einlesen von HTML-Dateien.

▶ **Textdateien**

Durch den Befehl `loadVariablesNum("http://Server.de/Verzeichnisse/datei.txt", 0);` lesen Sie Daten aus einer Datei ein. Der Dateinamen oder die Dateiendung ist dabei unwichtig. Sie muss lediglich einer gewissen Syntax entsprechen und sich auf demselben Server bzw. in derselben Subdomain befinden. Flash setzt die entsprechenden Variablen durch das Einlesen der Datei in den entsprechenden Level oder Movie. o bedeutet, dass die Variablen in `_Level0 (_root)` eingeladen werden. Sie können die Variablen aber auch direkt in einen Film einladen, um so eine bessere Übersicht über die Variablen zu erhalten.

```
loadVariables ("http://www.Server.de/Name.txt", "instanzname");
```