Autodesk®
Scaleform®

擴展 ActionScript 參考

本文檔描述了 Scaleform 4.2 中的擴展 ActionScript 2.0 的相關內容。

作者： Artem Bolgar
版本： 4.03
最新修訂： 2012 年 9 月 17 號

Autodesk®
GAMEWARE

Copyright Notice

Autodesk® Scaleform® 4.2

Autodesk Scaleform 聯繫方式：

文檔	擴展 ActionScript 2.0 參考
地址	Autodesk Scaleform Corporation 6305 Ivy Lane, Suite 310 Greenbelt, MD 20770, USA
網站	www.scaleform.com
郵箱	info@scaleform.com
電話	(301) 446-3200
傳真	(301) 446-3199

目錄

1 介紹

Autodesk Scaleform SDK 是一個結構簡潔、性能高效、功能豐富的 Adobe® Flash 向量圖形引擎，它建立於淨室工具，對控制臺和 PC 遊戲開發者特別有用。Scaleform 整合了應用廣泛的視覺化創作工具，如 Adobe® Creative Suite®，具備先進的硬體圖形加速功能，滿足了前沿遊戲開發者的需求。

由於 Flash 主要爲 Web 應用，而不是遊戲或者其他開發應用所設計，其功能在某些方面非常有限，如聚焦手柄、滑鼠支援、文本域支援和 IME 輸入支援。Scaleform 通過爲 ActionScript 類增加“擴展”，來改進這些方面的基本功能。爲使 Scaleform Player 中擴展功能有效，必須將`_global.gfxExtensions` 設置爲 `true`。

```
_global.gfxExtensions = true;
```

在大多數情況下，開發者在 FLA 文件的第一幀中就加入擴展功能有效聲明。如果未設置該變數，儘量與 Flash 保持最高等級的相容，Scaleform Player 將忽略所有擴展功能的引用。性。

開發者應該清楚本文檔中所描述的擴展功能在標準的 Flash® Player 中是不能使用的，只在 Scaleform 中有效。每項擴展功能對應一個 Scaleform 版本號，來區分增加擴展的播放器 SDK 發行號。在擴展功能版本號之前的 Scaleform Player 版本中將不支援。

儘管 Scaleform 盡最大力量去維持不同發行版本中擴展 API 函數的支援性和一致性，但是我們保留在將來 Scaleform 發行版本中修改、重命名或者移除擴展 API 函數的權利。如果有較大改動，我們將在主發行版本中體現出來，並儘早發出通告。

2 擴展滑鼠類

除了支援完整的標準滑鼠方法類，Scaleform 引進了一些其他的擴展功能，可以跟蹤多種類型滑鼠遊標並可識別 RIGHT, MIDDLE 和其他的滑鼠事件（至今已有 16 項）。本章首先簡單描述基本滑鼠功能，然後詳細描述了 Scaleform 滑鼠擴展功能。如需滑鼠物件更詳細的資料，建議開發者參考 Flash Mouse Class 文檔。

2.1 多類型遊標支援

在有些應用平臺中，如 Wii™遊戲控制臺，需要支援多種類型滑鼠遊標。Scaleform 3.2 中最多支援 4 中滑鼠遊標。GFx::Movie::SetMouseCursorCount(unsigned n)方法函數用來設置支援的遊標數。GFx::Movie::HandleEvent with GFx::MouseEvent 方法作爲將滑鼠事件賦給 Scaleform 內核的物件參數。GFx::MouseEvent 和 GFx::MouseCursorEvent 構造器具有擴展的參數來指派以零點爲基準的滑鼠索引。

2.2 擴展按鈕事件

滑鼠按鍵事件如 onRollOver, onRollOut, onDragOver, onDragOut, onPress, onRelease, onReleaseOutside 在 Scaleform 擴展有效時接收一個或兩個參數。第一個參數爲產生事件滑鼠基於零點基準的索引，從而，若索引號爲 1 的滑鼠遊標產生 onPress 事件，則該參數值爲 1：

```
mc.onPress = function(mouseIdx:Number)
{
      if (mouseIndex == 0)
            . . .
      else if (mouseIndex == 1)
            . . .
      . . .
}
```

onPress 和 onRelease 的第二個參數是數位屬性,指明事件是由滑鼠/游標還是鍵盤所引起。如果是鍵盤,則值為 -1,而如果是滑鼠/游標,則值為 0。這個屬性對於確定事件的起源非常有用：

```
mc.onPress = function(mouseIdx:Number, keyboardOrMouse:Number)
{
      if (keyboardOrMouse == 0)
            . . .
      else
```

```
            . . .
      . . .
}
```

參數 2 爲 onRollOver/Out 和 onDragOver/Out 事件選項。這些參數指定在相同文字上的 rollover/dragover 嵌套事件索引。如果這些功能處理參數未定義，則 onRollOver/onRollOut 和 onDragOver/onDragOut 事件對中只能觸發一次：onRollOver/onDragOver 只在遊標移到字元位置時觸發，而 onRollOut/onDragOut 在遊標最後離開時觸發。如果定義了處理這些事件的參數 2，則可產生嵌套的 onRollOver/onRollOut 和 onDragOver/onDragOut 事件（獨立於遊標），該參數則描述了基於零點位置的嵌套序號：初始時的 onRollOver/onDragOver 事件參數 2 爲 0；若第二個遊標滾過相同字元，則第二個 onRollOver/onRollOut 事件觸發並且第二個參數設置序號爲 1。如果任何一個遊標離開該字元，onRollOut/onDragOut 事件將被觸發並將參數 2 設置爲 1；最後的 onRollOut/onDragOut 事件觸發並設置參數 2 爲 0。

因此，參數 2 的定義改變了 onRollOver/onRollOut 和 onDragOver/onDragOut 事件的觸發方式。在參數 2 被定義時，用戶需要關注嵌套事件。

```
mc.onRollOver = function(mouseIdx:Number, nestingIdx:Number)
{
      // 只在 nestingIdx == 0 時做翻滾動作
      if (nestingIdx == 0)
            doOverAnimation();
      . . .
}
mc.onRollOut = function(mouseIdx:Number, nestingIdx:Number)
{
      //只在 nestingIdx == 0 時做翻滾動作
      if (nestingIdx == 0)
            doOutAnimation();
      . . .
}
```

注意，傳統事件如 RollOver，RollOut 等不提供擴展參數，因此不能區分被哪個遊標所觸發。從而不推薦在多滑鼠遊標中使用此類事件。
為了支援除用於 onPress、onRelease 等的左按鈕之外的滑鼠按鈕,增加了按鈕事件的新輔助版本：

- onPressAux
- onReleaseAux
- onReleaseOutsideAux
- onDragOver
- onDragOut

如果在事件目標上定義這些輔助事件處理常式,它們就會被調用。只針對包含 index != 0 (0 指數是指滑鼠左按鈕)的按鈕調用它們。始終只針對滑鼠左按鈕調用一般處理常式,而不管是否存在輔助事件處理常式。這些處理常式還具有與其標準副本(本節前面已定義)相同的函數簽名。不過,輔助事件處理常式為按鈕指數提供一個額外參數。

```
mc.onPressAux = function(mouseIdx:Number, keyboardOrMouse:Number,buttonIdx:Number)
{
      if (buttonIdx == 1)  // Right mouse button
            . . .
      . . .
}
```

2.3 滑鼠類事件

在 ActionScript 中可安裝滑鼠事件探測器來接收滑鼠移動、左鍵按鍵和滑鼠滾輪事件。在 Flash 中這些事件通常通過使用 Mouse.addListener 方法來安裝一個滑鼠事件探測物件來實現如下方法：

- `onMouseMove  = function() { }`
- `onMouseDown  = function() { }`
- `onMouseUp    = function() { }`
- `onMouseWheel = function(delta : Number, targetPath : String) { }`

安裝了探測物件後，當任何時候事件發生時相關的方法就會被調用。在 Flash 中 OnMouseDown 和 onMouseUp 方法只被滑鼠左鍵調用，沒有其他的公共介面來接收滑鼠事件。

爲應對這個限制（也爲支援多滑鼠遊標），Scaleform 擴展了這些方法調用，當 `_global.gfxExtensions` 變數設置爲真時，使其能夠獲得擴展參數。新的函數特性如下：

- `onMouseDown = function(button : Number, [targetPath : String],`
  `[mouseIdx : Number], [x : Number], [y : Number],`
  `[dblClick : Boolean]) { }`
- `onMouseUp = function(button : Number, [targetPath : String],`
  `[mouseIdx : Number], [x : Number], [y : Number]) { }`

當分配的滑鼠偵測函數獲得至少一個擴展參數，Scaleform 將調用這個偵測函數對應的方法來 處理 RIGHT, MIDDLE 和以及其他滑鼠事件，使得邏輯上能夠偵測到此類事件。LEFT，RIGHT， MIDDLE 對應的數值分別爲 1，2，3。當前最多支援 16 種事件。爲保持相容，若函數未定義參數，只調用 LEFT 滑鼠按鈕事件，在 Flash 中也相同。以下爲裝載此類處理功能的例子：

```
var mouseListener:Object = new Object;

mouseListener.onMouseDown = function(button, target)
{
   trace("mouseDown - button '" + button +
         "'  target = '" + target + "'");
}
mouseListener.onMouseUp = function(button, target)
{
   trace("mouseUp - button '" + button +
         "'  target = '" + target + "'");
}

Mouse.addListener(mouseListener);
```

新產生的滑鼠 down/up 控制碼至少包含兩個參數：按鍵和位置。參數一代表按鍵，描述滑鼠按鍵；與滑鼠 Mouse ["LEFT"], Mouse ["RIGHT"]和 Mouse ["MIDDLE"]擴展常數相比較解釋其含義（這裏用 Mouse["LEFT"] 來代替 Mouse.LEFT 表示方法是爲了與 ActionScript 2.0 編譯器相容）。參數二爲位置，提供滑鼠按下時遊標下面物件的頂部位置路徑資訊；如沒有此類路徑資訊該參數可不作定義。

在 Flash 中只可調用滑鼠左鍵按鈕控制碼，所有擴展參數都“未定義”。把“未定義”值看成是 LEFT 滑鼠按鈕，確保與 Flash player 相容。

爲支援多類型滑鼠遊標需要以下擴展參數：

- `onMouseMove = function([mouseIdx : Number], [x : Number], [y : Number]) { }`
- `onMouseDown = function(button : Number, [targetPath : String], [mouseIdx : Number], [x : Number], [y : Number],[dblClick : Boolean]) { }`
- `onMouseUp   = function(button : Number, [targetPath : String], [mouseIdx : Number], [x : Number], [y : Number]) { }`
- `onMouseWheel = function(delta : Number, targetPath : String, [mouseIdx : Number], [x : Number], [y : Number]) { }`

參數 `mouseIdx` 包含了觸發事件的遊標基於零點基準的索引號。`x` 和 `y` 參數則包含了遊標位置（_root 空間）。`onMouseDown` 控制碼的 `dblClick` 參數表示發生一個雙擊事件。這種情況下，參數值爲 `true`。

2.4 改變滑鼠遊標

Flash 中有三種類型的滑鼠遊標：

- 箭頭 - 遊標位於普通物件之上。
- 手形 - 在 useHandCursor 屬性爲真時，遊標位於按鈕和鏈結上。
- I 形 – 遊標位元於文本域上。

Flash 中不無法偵測到滑鼠遊標的變化，Scaleform 提供了 C++ API 函數和 ActionScript 滑鼠擴展類來管理自定義遊標。在 C++中，可以使用 `GFx::StateBag::SetUserEventHandler` 函數安裝事件控制碼，當滑鼠遊標改變時觸發。使用這個控制碼，如 FxPlayer.cpp 中所示，既可改變遊戲引擎所繪製的遊標，也可改變系統遊標。同時，也可以選擇完全採用 ActionScript 實現自定義遊標，能體現出其動畫和藝術控制優勢。SDK 中提供了該方法的一個實例文 *Mouse.fla/swf*。

滑鼠實例實現自定義遊標的方法爲將其當作在場景中的一個簡單視頻剪輯，稱爲‘Cursor _M’。遊標包括了不同的幀，不同幀中分別爲“手形”，“箭頭”和“I-形”；可以通過移動時間線來改變遊標圖示。當偵測到滑鼠移動時，視頻剪輯的_x 和_y 座標隨著滑鼠而改變。

爲了改變位於物件上面的遊標類型，滑鼠實例採用了 Scaleform 擴展函數 `Mouse.setCursorType`。該函數當遊標需要改變時調用 cursorType 和 mouseIndex 參數，允許遊標改變動作被偵測到。通過比較滑鼠參數(Mouse["ARROW"]，Mouse["HAND"] and Mouse["IBEAM"])的 cursorType 類型並適當得改變遊標幀來展示滑鼠實例。

下面爲處理滑鼠實例中多類型自定義遊標的源代碼。建議查看實例文件獲得詳細的資訊。

```
_global.gfxExtensions = 1;
// 隱藏系統遊標。
Mouse.hide();

var mouseListener:Object = new Object;
mouseListener.onMouseMove = function(mouseIdx, x,y)
{
    if (mouseIdx == undefined)
        mouseIdx = 0;
    if (x == undefined)
    {
        x = _xmouse;
        y = _ymouse;
    }
    var cmc = eval("_root.Cursor_M"+(mouseIdx+1));

    cmc._x = x;
    cmc._y = y;
}
Mouse.addListener(mouseListener);
```

```
Mouse["setCursorType"] = function(cursorType, mouseIdx)
{
   if (mouseIdx == undefined)
      mouseIdx = 0;
   var cmc = eval("_root.Cursor_M"+(mouseIdx+1));
   switch(cursorType)
   {
      case Mouse["HAND"]:
         cmc.Cursor.gotoAndPlay("hand");
         break;
      case Mouse["ARROW"]:
         cmc.Cursor.gotoAndPlay("arrow");
         break;
      case Mouse["IBEAM"]:
         cmc.Cursor.gotoAndPlay("ibeam");
         break;
      default: return;
   }
}

// 運用 Mouse.show 和 Mouse.hide 函數
// 將對遊標產生影響。
ASSetPropFlags(Mouse, "show,hide", 0, 7);

Mouse.show = function(mouseIdx)
{
    if (mouseIdx == undefined)
        mouseIdx = 0;
    var cmc = eval("_root.Cursor_M"+(mouseIdx+1));
    cmc._visible = true;
}
Mouse.hide = function(mouseIdx)
{
    if (mouseIdx == undefined)
        mouseIdx = 0;
    var cmc = eval("_root.Cursor_M"+(mouseIdx+1));
    cmc._visible = false;
}
```

2.5 ActionScript 擴展滑鼠類

Scaleform 中 ActionScript 擴展滑鼠類中增加了一些靜態方法和屬性。ActionScript 1.0 中可以直接引用擴展函數，如 `Mouse.setCursorType(...)`。而在 ActionScript 2.0 沒有如此方便，直接引用將導致異常，爲減輕 ActionScript 2.0 編譯器負擔，需使用不用的語法，如:`Mouse["setCursorType"](..)`。

getButtonsState() 靜態方法

```
public function getButtonsState(mouseIndex : Number) : Number
```

Scaleform 版本： 2.2

返回滑鼠按鈕狀態。返回的值爲一個位遮罩值，每個位元代表一個滑鼠按鈕序號。但對應的位元爲 1 則代表該按鈕已被按下。

參數

`mouseIndex : Number` – 基於零點的滑鼠索引

getTopMostEntity()靜態方法

```
public function getTopMostEntity([mouseIndex : Number]) : Object
public function getTopMostEntity(testAll : Boolean,
                                 [mouseIndex : Number]) : Object
public function getTopMostEntity(x : Number, y : Number) : Object
public function getTopMostEntity(x : Number, y : Number, testAll: Boolean) : Object
```

Scaleform 版本： 1.2.34, 從 2.2 版本開始支援多類型滑鼠遊標

靜態方法返回滑鼠遊標下的目標字元或者指定物件。此方法在有按鈕控制碼設置（如 onPress, onMouseDown, onRollOver 等）和無按鈕控制碼設置的字元並不相同。這樣可以區分出哪些字元無滑鼠事件控制碼需要處理。

此方法在 `MovieClip.hitTest` 正好起到反向作用， `hitTest` 檢查 x 和 y 是否對應在目標物件範圍之內，而 `getTopMostEntity` 返回對應於 x 和 y 座標的實際物件。從而，`Mouse.getTopMostEntity(x, y).hitTest(x, y, true) == true`。

參數

`mouseIndex : Number` –基於零點的滑鼠索引

`testAll : Boolean` – 指示按鈕控制碼（false）字元或任何字元（false）。如該參數並沒指定，`getTopMostEntity` 默認其爲‘true’。

`x : Number, y : Number` – 字元查找位置

示例:

```
var listenerObj = new Object;
```

```
listenerObj.onMouseMove = function()
{
   var target = Mouse["getTopMostEntity"]();
   trace("Mouse moved, target = "+target);
}

Mouse.addListener(listenerObj);
var target = Mouse["getTopMostEntity"](480, 10);
trace("The character at the (480, 10) is " + target);
```

getPosition 靜態方法

```
public function getPosition(mouseIndex : Number) : flash.geom.Point
```

Scaleform 版本： 2.2

該方法返回對應的滑鼠遊標位置，基於_root 位置空間。返回的值爲一個 flash.geom.Point 實例。

參數

`mouseIndex : Number` – 基於零點的滑鼠索引

setCursorType() 靜態方法

```
public function setCursorType(cursorType : Number, [mouseIndex : Number]) : void
```

Scaleform 版本： 1.2.32

本靜態方法根據參數 cursorType 改變滑鼠遊標。參閱“改變滑鼠遊標”獲得詳細資訊。

參數

`cursorType : Number` – 顯示遊標類型， Mouse.ARROW, Mouse.HAND, Mouse.IBEAM.

`mouseIndex : Number` –基於零點的滑鼠索引

3 擴展按鈕類

hitTestDisable 屬性

`hitTestDisable:Boolean  [讀-寫]`

Scaleform 版本： 2.1.51

當本屬性設置爲 `true` 時，`MovieClip.hitTest` 函數在按鈕點擊時忽略該按鈕。並且對應於該按鈕的所有滑鼠事件將無效。

預設值爲 `false`

可參考：

```
TextField.hitTestDisable
MovieClip.hitTestDisable
```

topmostLevel 屬性

`topmostLevel:Boolean  [讀-寫]`

Scaleform 版本： 2.1.50

如果本屬性設置爲 `true`，則此字元將在其他字元頂層顯示，不管其層疊深度如何。這在自定義滑鼠遊標中當遊標需要在所有物件頂層顯示時候有用。預設值爲 false

在將幾個特性標識為 “topmostLevel” 時,繪圖命令如下所示:

- 對於 Scaleform 3.0.71 而言,將特性標識為 “topmostLevel” 的繪圖命令取決於將這一屬性設置為“真”的命令(因此,首先被標識為“置頂”的特性將首先繪出);
- 從 Scaleform 3.0.72 開始,繪圖命令都是相同的,因為它不再標識“置頂”特性,即將物件 A 繪製在物件 B 下麵,置頂後物件 A 仍然位於物件 B 下方,而無論使用何種命令將 “topmostLevel” 的屬性設置為“真”。

注釋:一旦某一屬性被標識為 “topmostLevel”,swapDepth ActionScript 功能將不會對這一特性產生任何影響。

可參考：

```
TextField.topmostLevel
MovieClip.topmostLevel
```

focusGroupMask 屬性

```
focusGroupMask : Number
```

Scaleform 版本： 3.3.84

此屬性將一個位元遮罩設置為一個舞臺人物及其全部 (ALL) 子項。此位元遮罩將焦點組所有權指定給該角色,意思是只有位元遮罩中表明的控制器才能將焦點移動到角色內並在其中移動焦點。可以使用 setControllerFocusGroup 擴展方法將焦點組與控制器相關聯。

例如,我們假設“按鈕 1”(button1) 只能作為控制器 0 的焦點,而“電影剪輯 2”(movieclip2) 只能作為控制器 0 和 1 的焦點。要實現這一行為,請將焦點組與控制器相關聯：

```
Selection.setControllerFocusGroup(0, 0);
Selection.setControllerFocusGroup(1, 1);

button1.focusGroupMask = 0x1; // bit 0 – focus group 0
movieclip2.focusGroupMask = 0x1 | 0x2 // bits 0 and 1 – focus groups 0 and 1
```

“focusGroupMask” 位元遮罩可以設置為父電影剪輯 (parent movieclip)。這將把遮罩值傳播到所有其子項。

4 MovieClip 擴展類

hitTest ()方法

```
public hitTest(x:Number, y:Number, [shapeFlag:Boolean],
               [ignoreInvisibleChildren:Boolean]):Boolean
```

Scaleform 版本： 2.1.51

標準的 3 參數 hitTest 還擁有一個布林參數擴展項，指示是否忽略或不顯示其子剪輯。Flash 中 hitTest 的默認爲返回“true”，甚至在 x，y 位置處於可視的子剪輯也同樣如此（如_visible 屬性設爲 false；子剪輯_alpha 設爲 0 時當作不可視處理）。

參數

`ignoreInvisibleChildren:Boolean` – 應該設置爲 true 忽略所有可視子剪輯。

hitTestDisable 屬性

```
hitTestDisable:Boolean  [讀-寫]
```

Scaleform 版本： 1.2.32

當本屬性設置爲 `true` 時，`MovieClip.hitTest` 函數在按鈕點擊時忽略該按鈕。並且對應於該按鈕的所有滑鼠事件將無效。

預設值爲 `false`。

可參考：

`TextField.hitTestDisable`
`Button.hitTestDisable`

noAdvance 屬性

```
noAdvance : Boolean
```

Scaleform 版本：2.1.52

本屬性設置爲 true 時，關閉視頻剪輯和所有子剪輯的前進動作。這可以改進性能。
注釋，Flash 中視頻剪輯通常有前進動作（按時間線執行動畫，調用幀中的 ActionScript 等）。因此，在 Scaleform 和 Flash 中設置屬性爲“true”可導致不同的效果。

可參考：

```
_global.noInvisibleAdvance
```

topmostLevel 屬性

```
topmostLevel:Boolean  [讀-寫]
```

Scaleform 版本： 2.1.50

若本屬性設置爲 true，則此字元將在其他字元頂層顯示，不管其層疊深度如何。這在自定義滑鼠遊標中當遊標需要在所有物件頂層顯示時候有用。預設值爲 false。

在將幾個特性標識為 “topmostLevel” 時,繪圖命令如下所示:

- 對於 Scaleform 3.0.71 而言,將特性標識為 “topmostLevel” 的繪圖命令取決於將這一屬性設置為“真”的命令(因此,首先被標識為“置頂”的特性將首先繪出);
- 從 Scaleform 3.0.72 開始,繪圖命令都是相同的,因為它不再標識“置頂”特性,即將物件 A 繪製在物件 B 下麵,置頂後物件 A 仍然位於物件 B 下方,而無論使用何種命令將 “topmostLevel” 的屬性設置為“真”。

注釋:一旦某一屬性被標識為 “topmostLevel”,swapDepth ActionScript 功能將不會對這一特性產生任何影響。

可參考

```
TextField.topmostLevel
Button.topmostLevel
```

rendererString 屬性

```
rendererString:String  [讀-寫]
```

Scaleform 版本： 2.2.55

本屬性允許在任何 MovieClip 實例中通過 ActionScript 發送自定義變數到渲染器。如果本屬性置位元，字串值將作爲用戶資料傳遞給渲染器。

無預設值。

示例：

```
myMovieInstance.rendererString = "SHADER_Blur"
```

rendererFloat 屬性

`rendererFloat:Number  [讀-寫]`

Scaleform 版本： 2.2.55

本屬性允許在任何 MovieClip 實例中通過 ActionScript 發送自定義變數到渲染器。如果本屬性置位元，浮點值將作爲用戶資料傳遞給渲染器。

無預設值。

示例：

```
myMovieInstance.rendererFloat = 4; // 例如 C++ 枚舉值
```

disableBatching 屬性

`disableBatching:Boolean`

Scaleform 版本: 4.0.17

此屬性禁用針對自訂繪圖的網格生成批次處理。

focusGroupMask 屬性

`focusGroupMask : Number`

Scaleform 版本： 3.3.84

此屬性將一個位元遮罩設置為一個舞臺人物及其全部 (ALL) 子項。此位元遮罩將焦點組所有權指定給該角色,意思是只有位元遮罩中表明的控制器才能將焦點移動到角色內並在其中移動焦點。可以使用 setControllerFocusGroup 擴展方法將焦點組與控制器相關聯。

例如,我們假設“按鈕 1”(button1) 只能作為控制器 0 的焦點,而“電影剪輯 2”(movieclip2) 只能作為控制器 0 和 1 的焦點。要實現這一行為,請將焦點組與控制器相關聯：

```
Selection.setControllerFocusGroup(0, 0);
Selection.setControllerFocusGroup(1, 1);

button1.focusGroupMask = 0x1; // bit 0 – focus group 0
movieclip2.focusGroupMask = 0x1 | 0x2 // bits 0 and 1 – focus groups 0 and 1
```

“focusGroupMask” 位元遮罩可以設置為父電影剪輯 (parent movieclip)。這將把遮罩值傳播到所有其子項。

5 擴展 NetStream 類

Scaleform 添加功能函數到 NetStream 類，該類可以添加字幕和音頻軌道到視頻文件。

onMetaData 事件控制碼

```
onMetaData = function(info:Object) { }
```

Scaleform 版本： 3.0.63

事件控制碼在視頻文件播放時接收資訊，包括 Scaleform 中的兩個附加物件屬性。這些屬性用來獲得視頻文件自帶的字幕和音頻軌道資訊。

物件傳遞給 `onMetaData` 事件控制碼，包含了兩個附加的唯讀屬性。

1. `audioTracks` – 視頻文件音頻軌道編碼描述物件組
   `audioTracks` 序列中的每個物件包含以下屬性：
   - `channelsNumber` – 音頻軌道通道數 (1–單聲道, 2–身歷聲, 6–5.1 環繞聲)。
   - `totalSamples` – 音頻軌道中聲音取樣數。
   - `trackIndex` – 軌道序號，本屬性用來選擇音頻軌道，只需要將其賦給 `NetStream.audioTrack` 屬性即可 (參見下面實例)。
   - `sampleRate` – 聲音取樣率。

2. `subtitleTracksNumber` – 視頻文件中包含的字幕通道數。

音頻通道屬性

```
audioTrack:數量  [讀-寫]
```

Scaleform 版本： 3.0.63

本屬性用來設置/獲取當前視頻文件播放音頻的軌道。

例子：

```
var ns:NetStream = new NetStream(nc);
var audioTracks;

ns.onMetaData = function(info:Object)
```

```
{
      audioTracks = info.audioTracks;
}

if (audioTracks != undefined && audioTracks.length > 0)
      ns.audioTrack = audioTracks[0].trackIndex;
```

字幕通道屬性

```
subtitleTrack:Number  [讀-寫]
```

Scaleform 版本： 3.0.63

該屬性可以設置和獲得當前字幕通道。關閉字幕將屬性設置爲 0。

onSubtitle 事件控制碼

```
onSubtitle = function(msg:String) {}
```

Scaleform 版本： 3.0.63

該事件控制碼當字幕消息準備完畢需要被播放時調用。

例子：

```
var ns:NetStream = new NetStream(nc);
var subtitlesNumber = 0;

ns.onMetaData = function(info:Object)
{
      subtitlesNumber = info. subtitleTracksNumber;
}

ns.onSubtitle = function(msg:String)
{
      sbTextField.text = msg;
}
if (subtitlesNumber > 0)
      ns.subtitleTrack = 1;
```

setNumberOfFramePools 函數

```
public function setNumberOfFramePools(pools : Number) : Void
```

Scaleform 版本： 3.0.68

該函數用來設置內部視頻緩存區數量。視頻緩存區用來保存視頻編碼輸出內容，被稱爲“幀池”。當用來編碼的 CPU 資源負載不均衡，擁有較多幀池可以讓播放畫面更加平滑。預設值爲 1。

該方法應該在視頻開始播放前被調用（例如，在函數 NetStream.play()前被調用）。

setReloadThresholdTime 函數

```
public function setReloadThresholdTime(reloadTime : Number) : Void
```

Scaleform 版本： 3.0.68

該函數設置重載時間，以秒爲單位。當輸入緩存區的資料量小於重載限度，則 Scaleform 視頻庫請求讀取下一個文件。時間限度根據視頻文件比特率和重載時間設置自動判定。重載時間預設值爲 0.8 秒。

在視頻播放時讀取遊戲資料，使用本方法可以減少搜索請求。例如，當設置 Netstream 緩存時間（NetStream.setBufferTime() 方法）爲 2 秒，設置時間限度爲 1 秒，遊戲資料可以連續讀取 1 秒。但是遊戲資料讀取時間應該不大於重載時間內。當遊戲資料過大，可能需要分塊讀取。如果遊戲資料讀取未能在重載時間內完成，由於視頻資料緩存區爲空，則視頻播放將“卡片”。

該方法應該在視頻開始播放前被調用（例如，在 NetStream.play()函數前調用）。

6 擴展選擇類

選擇類，在其他函數中允許用於管理文本域，視頻剪輯和按鈕的輸入焦點。

Scaleform 擴展了選擇類的焦點功能函數。ActionScript 1.0 中可以直接引用選擇類擴展函數，如 `Selection.captureFocus()`。而在 ActionScript 2.0 沒有如此方便，直接引用將導致異常，爲減輕 ActionScript 2.0 編譯器負擔，需使用不用的語法，如：`Selection["captureFocus"]()`。

alwaysEnableArrowKeys 靜態屬性

```
alwaysEnableArrowKeys : Boolean
```

Scaleform 版本： 1.2.34

本靜態屬性允許箭頭鍵改變焦點，甚至在“`_focusrect`”屬性設置爲 `false` 時也是如此（應用於焦點捕獲時）。默認情況下，在黃色焦點通過設置“`_focusrect = false`”被禁止時 Flash 不允許使用箭頭鍵來改變焦點；要改變這個行爲，需將 `alwaysEnableArrowKeys` 屬性設置爲 “`true`” 。

示例：

```
Selection["alwaysEnableArrowKeys"] = true;
```

alwaysEnableKeyboardPress 靜態屬性

```
alwaysEnableKeyboardPress : Boolean
```

Scaleform 版本： 3.0.63

該靜態屬性允許通過按下空白鍵或回車鍵觸發 onPress / onRelease 事件，甚至在 `focusrect` 屬性設置爲 `false` 時也可以（在焦點被捕獲時應用）。默認情況下，若黃色焦點框通過 `focusrect = false` 設置禁止，則 `Flash` 不允許通過鍵盤控制按鈕；爲改變此特性，設置 `alwaysEnableKeyboardPress` 屬性爲 `true`。

例如：

```
Selection["alwaysEnableKeyboardPress"] = true;
```

captureFocus() 靜態方法

```
public function captureFocus([doCapture:Boolean, controllerIdx:Number]) : void
```

Scaleform 版本： 1.2.34

本靜態屬性可編程實現捕捉或釋放鍵盤焦點。

captureFocus 中 doCapture 設置爲 true（或者無任何參數）可捕獲鍵盤焦點，並允許方向鍵來改變焦點，效果如同首次按下 Tab 鍵。
captureFocus 中 doCapture 設置爲 false 則釋放鍵盤焦點，效果如同焦點有效時滑鼠移動。

參數

doCapture:Boolean – 可選擇，表示鍵盤焦點需要捕獲還是釋放。如果這個參數被忽略則 captureFocus 與該參數設置爲 true 時行爲相同。

controllerIdx:Number –可選,表示哪個鍵盤/控制器用於捕獲操作。預設情況下,使用控制器 0。

示例：

```
Selection["captureFocus"](); // 與傳遞“true”相同
Selection["captureFocus"](false);
```

disableFocusAutoRelease 靜態屬性

```
disableFocusAutoRelease : Boolean
```

Scaleform 版本： 1.2.34

本靜態屬性用來控制一個滑鼠動作是否需要釋放鍵盤焦點（Flash 中標準行爲爲需要釋放）。默認情況下，若焦點被捕獲，黃色矩形顯示（允許方向鍵來改變焦點），則滑鼠動作釋放焦點。當設置本屬性爲 true 時將禁止此動作。

示例：

```
Selection["disableFocusAutoRelease"] = true;
```

disableFocusKeys 靜態屬性

```
disableFocusKeys : Boolean
```

Scaleform 版本： 2.2.58

本靜態屬性禁止所有焦點鍵的處理（TAB, Shift-TAB 和方向鍵），因此，用戶可以實現其自身焦點鍵管理。

示例：

```
Selection["disableFocusKeys"] = true;
```

可參考：

```
moveFocus()
```

disableFocusRolloverEvent 靜態屬性

```
disableFocusRolloverEvent : Boolean
```

Scaleform 版本： 2.0.37

本屬性用來禁止焦點被鍵盤改變時觸發 rollover/out 事件。設置本屬性爲 `true` 可禁止此行爲。

示例：

```
Selection["disableFocusRolloverEvent "] = true;
```

modalClip 靜態屬性

```
modalClip : MovieClip
```

Scaleform 版本： 2.2.58

本靜態屬性設置焦點管理視圖中的視頻剪輯爲“modal”剪輯。這意味著按鍵焦點（TAB,Shift-TAB 和方向鍵）只在特定視頻剪輯中才可以移動焦點矩形，如，只限制在視頻剪輯的“tabable”子項。

示例：

```
Selection["modalClip"] = _root.mc;
```

```
...
Selection["modalClip"] = undefined;
```

可參考：

```
disableFocusKeys
moveFocus()
```

moveFocus() 靜態方法

```
public function moveFocus(keyToSimmulate : String [, startFromMovie:Object,
                includeFocusEnabledChars : Boolean = false,
                controllerIdx :Number]) : Object
```

Scaleform 版本: 2.2.58

本靜態方法用來通過類比鍵盤按鍵動作來移動焦點矩形，這些按鍵包括 TAB, Shift-TAB 或方向鍵。本方法與 `disableFocusKeys` 和 `modalClip` 屬性配合使用，可以實現自定義焦點管理功能。

參數

`keyToSimmulate:String` – 類比按鍵名稱：“up”, “down”, “left”, “right”, “tab”, “shifttab”。

`startFromMovie:Object` – 可選參數指定字元；moveFocus 將使用其將當前焦點作爲起始點。該屬性可能爲 null 或未定義，意爲當前獲得焦點的字元爲起始指標；這在制定第三個參數選項時或許有用。

`includeFocusEnabledChars:Boolean` – 可選標記，設置允許 moveFocus 用在只有 `focusEnabled` 可設置的字元上或者只有 `tabEnabled / tabIndex` 屬性可設置的字元上。如果該標記被指定或設置爲 `false`，則只有具有 `tabEnabled / tabIndex` 屬性設置的字元可以用與焦點移動。

`controllerIdx:Number` –可選,表示哪個鍵盤/控制器用於捕獲操作。預設情況下,使用控制器 0。

示例

```
Selection["moveFocus"]("up");
Selection["moveFocus"]("tab", _root.mc);
Selection["moveFocus"]("tab", _root.mc, true);
Selection["moveFocus"]("tab", null, true);
```

返回值

返回新聚焦字元，或無法識別字元返回未定義內容。

可參考：

disableFocusKeys
modalClip

findFocus()靜態方法

```
public function findFocus(keyToSimmulate : String [, parentMovie:Object, loop :
                Boolean, startFromMovie:Object, includeFocusEnabledChars : Boolean,
                controllerIndex : Number]) : Object
```

Scaleform 版本： 3.3.84

此靜態方法用來通過以類比方式按下列各鍵之一來查找下一個焦點項:TAB、Shift-TAB 或方向鍵。此方法可與 disableFocusKeys 和 setModalClip/getModalClip 擴展一起用來實現自訂焦點管理。

參數

keyToSimulate:String-要類比的鍵的名稱："向上"、"向下"、"向左"、"向右"、"選項卡上"、"shifttab"。

parentMovie -用作模態剪輯的電影剪輯。焦點項搜索只在此剪輯的子項內執行。可以是“空”(Null)。

loop - 迴圈焦點的布林旗標。例如,如果當前焦點項位於底部,而鍵為“向下”鍵,則 findFocus 要麼返回 “null”(如果此旗標為 “false”(假)的話),要麼返回最上面的焦點項(如果旗標為 “true”(真)的話)。

startFromMovie:Object –可選參數,指定 findFocus 將使用的人物,而非作為起點的當前焦點項。此屬性可能是 “null”或 “ undefined”(未定義),意思是將當前焦點人物用作起點。

includeFocusEnabledChars:Boolean –可選旗標,允許 moveFocus 移動到只設置 focusEnabled 屬性的人物以及設置 tabEnabled/tabIndex 屬性集的人物上。如果沒把旗標指定或設置為 false(假),則只有設置了 tabEnabled/tabIndex 屬性的人物將參與焦點移動。

controllerIndex -正在操縱焦點的控制器的一個零基指數。這可與焦點組一起用來提供多控制器焦點支援。

示例：

```
var a = Selection["findFocus"]("up");
```

返回

返回要作為焦點的下一個人物,或者返回 null --- 如果沒有找到該人物。

另請參見：

disableFocusKeys
setModalClip
getModalClip

setModalClip()靜態方法

```
public function setModalClip(modalClip : Object, controllerIndex : Number)
```

Scaleform 版本： 3.3.84

此靜態方法把指定的電影剪輯設置為用於焦點管理的“模態”剪輯。這意味著 TAB、Shift-TAB 和方向鍵將在所有 “tabable” 子項之間的指定電影剪輯內移動焦點。

参数

`modalClip` – 一个模态剪辑。
`controllerIndex` - 控制器的一个零基指数。

getModalClip()靜態方法

```
public function getModalClip(controllerIndex : Number) : Object
```

Scaleform 版本： 3.3.84

此靜態方法為指定控制器返回模態剪輯。

参数

`controllerIndex` -控制器的零基指數。

返回

一個模態剪輯,或者未定義 --- 如果沒有找到的話。

setControllerFocusGroup()靜態方法

```
public function setControllerFocusGroup (controllerIndex : Number, focusGroupIdx :
                Number) : Boolean
```

Scaleform 版本： 3.3.84

此靜態方法將 controllerIndex 指明的控制器與一個焦點組關聯。預設情況下,所有控制器都與焦點組 0 關聯,這意味著它們在用同一焦點。不過,可以使每個控制器都使用其各自的焦點。例如,如果兩個控制器都有單獨的焦點(在分屏使用情況下),那麼 `setControllerFocusGroup (1,1)` 將為控制器 1 創建一個單獨的焦點組。調用 `setControllerFocusGroup (1,0)` 將使控制器 0 和 1 再次共用同一焦點。

参数

`controllerIndex` – 控制器的零基指數。
`focusGroupIdx` –焦點組的零基指數。

示例：

```
Selection["setControllerFocusGroup"](0,0);
Selection["setControllerFocusGroup"](1,1);
Selection["setControllerFocusGroup"](2,1);
```

返回

返回 true --- 如果成功的話。

getControllerFocusGroup()靜態方法

```
public function getControllerFocusGroup (controllerIndex : Number) : Number
```

Scaleform 版本： 3.3.84

此靜態方法返回與指定控制器關聯的焦點組。

參數

`controllerIndex` -物理控制器的零基指數。

返回

焦點組的基於零的指數。

getFocusArray()靜態方法

```
public function getFocusArray(mc : Object) : Array
```

Scaleform 版本： 3.3.84

此靜態方法返回一組控制器指數,這些指數當前在指定電影剪輯/按鈕/文字方塊上擁有焦點。

參數

`mc` -一個電影剪輯、按鈕或文字方塊。

返回

基於零的指數(數位)組。

getFocusBitmask()靜態方法

```
public function getFocusBitmask(mc : Object) : Number
```

Scaleform 版本： 3.3.84

此靜態方法返回一個位元遮罩,其中每個位代表一個當前在指定電影剪輯/按鈕/文字方塊上有焦點的控制器。

參數

`mc` -一個電影剪輯、按鈕或文字方塊。

返回

控制器的一個位元遮罩。

getControllerMaskByFocusGroup()靜態方法

```
public function getControllerMaskByFocusGroup (focusGroupIdx :Number) :Number
```

Scaleform 版本： 3.3.84

此靜態方法返回一個位元遮罩,其中每個位代表一個與指定焦點組關聯的控制器。返回 setControllerFocusGroup 函數設置的狀態。

參數

`focusGroupIdx` -焦點組的一個指數。

返回

控制器的一個位元遮罩。

numFocusGroups 屬性

```
numFocusGroups : Number
```

Scaleform 版本： 3.3.84

返回焦點組的數目,通過調用 setControllerFocusGroup 函數來設置。如果焦點組 0 和 3 是活動的,則 numFocusGroups 將返回 2。

7 擴展 TextField 類

Scaleform 引入了衆多 TextField 擴展類，使得比 Flash 8 中的內嵌 TextField 類更加易用和強大。 有些爲複製了 Flash 9 中的內嵌 TextField 相同或相似的功能。增加的擴展功能是爲了更好地管理 IME、對齊、文本大小和外觀、文本濾鏡效果（如陰影，模糊等）、嵌入圖像、選擇和剪貼等操作。

7.1 般性功能

本章描介紹了擴展屬性和方法的一般性用途。

autoFit 屬性

```
autoFit:Boolean  [讀-寫]
```

Scaleform 版本： 2.0.39

設置 on/off 字體自動。

預設值爲 false。

appendText () 方法

```
public function appendText(newText:String):void
```

Scaleform 版本： 2.0.37

用 newText 參量將字串追加到文本結尾處。這種方法比用（+=）符號來附加字串更加有效（例如，`my_txt.text += appendingText`），在文本域中包含大量文本時特別有用。

注釋： 如果將某一樣式表格應用到文本欄位中,則本方法不適用。

參數

`newText:String` – 附加字串

可參考

`appendHtml()`

appendHtml () 方法

```
public function appendHtml(newHtml:String):void
```

Scaleform 版本： 2.0.37

用 newHtml 參量附加 HTML 文本到文本域結尾處。這種方法比用（+=）符號來附加更加有效（如 `txt.htmlText += moreHtml`）。位於 htmlText 屬性的+=追加符將生成 HTML 標識，附加新的 HTML 文本並對其進行解析。該函數增加了 HTML 解析功能，如只能從 newHtml 字串常量中解析出 HTML 文本（這也是在 newHtml 常量中 HTML 文本應該用特定格式表示的原因，這對於+=操作符並無要求）。此方法在文本域中包含大量文本時特別有用。

注釋： 如果將某一樣式表格應用到文本欄位中,則本方法不適用。

參數

`newHtml:String` – 追加到顯存文本的 HTML 字串。

可參考：

`appendText()`

caretIndex 屬性

```
caretIndex:Number  [讀-寫]
```

Scaleform 版本： 2.0.37

該屬性描述了插入位置（符號或者指標形式）的索引。如果文本域沒有獲得焦點，字元位置沒有被顯示出來（閃爍遊標形式），則表示上次獲得焦點的字元位置；如果文本域從沒獲得過焦點，則表示爲 0。該屬性包含的值與 `Selection.getCaretIndex()`方法函數返回值相同；但是，`caretIndex` 即使在文本域沒有獲得焦點的時候也可發揮作用，而 `Selection.getCaretIndex()`在此情況下則返回-1。

符號索引以零爲基準（因此，首位置爲 0）。

可參考：

`Selection.getCaretIndex()`

getCharBoundaries () 方法

```
public function getCharBoundaries(charIndex:Number): flash.geom.Rectangle
```

Scaleform 版本： 2.0.37

該方法返回字元邊框的矩形位置。注釋，該方法返回的矩形爲每個字元輪廓的靠前估略值。這意味著返回的矩形是一個大致的輪廓框（看下圖，紅色矩形描述了 ‘a’ 、 ‘w’ 和 ‘k’ 的輪廓）：

參數

`charIndex:Number` – 以零爲基準的字元位置索引值（如，首位置爲 0，第二個位置爲 1，依次類推）。

返回值

flash.geom.Rectangle – 矩形的 x，y 座標起始位置值，定義了字元的邊框。與文本域的空間位置相對應，如（0，0）點對應于文本域的左上角。

可參考：

`getExactCharBoundaries()`

getExactCharBoundaries () 方法

```
public function getExactCharBoundaries(charIndex:Number): flash.geom.Rectangle
```

Scaleform 版本： 2.0.37

該方法返回字元精確邊框的矩形位置。注釋，本方法返回的矩形框爲字元輪廓的準確大小（而`getCharBoundaries()`返回的是靠前估略的輪廓大小）。矩形的高度爲整個字串的整體高度。查看下圖：紅色矩形顯示了 ‘a’ 、 ‘w’ 和 ‘k’ 的精確邊框：

參數

charIndex:Number -以零爲基準的字元位置索引值（如，首位置爲 0，第二個位置爲 1，依次類推）。

返回值

flash.geom.Rectangle - 矩形的 x，y 座標起始位置值，定義了字元的邊框。與文本域的空間位置相對應，如（0，0）點對應于文本域的左上角。

可參考：

getCharBoundaries()

getCharIndexAtPoint () 方法

```
public function getCharIndexAtPoint(x:Number, y:Number): Number
```

Scaleform 版本： 2.0.37

本方法返回用 x 和 y 參數表示的基於零點的字元位置索引值。

參數

x:Number – 字元的 x 座標。

y:Number – 字元的 y 座標。

返回值

Number -以零爲基準的字元位置索引值（如，首位置爲 0，第二個位置爲 1，依次類推）。座標不位於任何字元上返回-1。

getFirstCharInParagraph () 方法

```
public function getFirstCharInParagraph(charIndex:Number): Number
```

Scaleform 版本： 2.0.37

該方法用 charIndex 參量返回段落中第一個字元的索引。

參數

charIndex:Number -以零爲基準的字元位置索引值（如，首位置爲 0，第二個位置爲 1，依次類推）。

返回值

Number – 以零點爲基準的段落中第一個字元索引。

getLineIndexAtPoint () 方法

```
public function getLineIndexAtPoint(x:Number, y:Number): Number
```

Scaleform 版本： 2.0.37

本方法返回用 x 和 y 參數表示的基於零點的字串位置索引值。

參數

`x:Number` -字元的 x 座標。
`y:Number` -字元的 y 座標。

返回值

Number - 以零爲基準的文本串索引值（如，第一行文本串爲 0，第二行字串爲 1，依次類推）。座標不位於任何字串上返回-1。

getLineLength() 方法

```
public function getLineLength(lineIndex:Number): Number
```

Scaleform 版本： 2.0.37

返回特定文本行包含的字元數。

參數

`lineIndex:Number` - 以零爲基準的文本行數量（如，第一行文本行爲 0，第二行文本行爲 1，依次類推）。

返回值

Number – 字元行中包含的字元數量。

getLineMetrics() 方法

```
public function getLineMetrics(lineIndex:Number): Object
```

Scaleform 版本： 2.0.43

返回一特定文本行中的格式資訊。返回的物件包括以下部分：

`ascent : Number`

以圖元爲單位元描述文本高度，即從基準文本行到頂行的高度。

`descent : Number`

以圖元爲單位元描述文本高度，即從基準文本行到底行的高度。

`height : Number`

以圖元爲單位元描述文本高度，選擇行高度（不需要全部文本空間）。

`leading : Number`

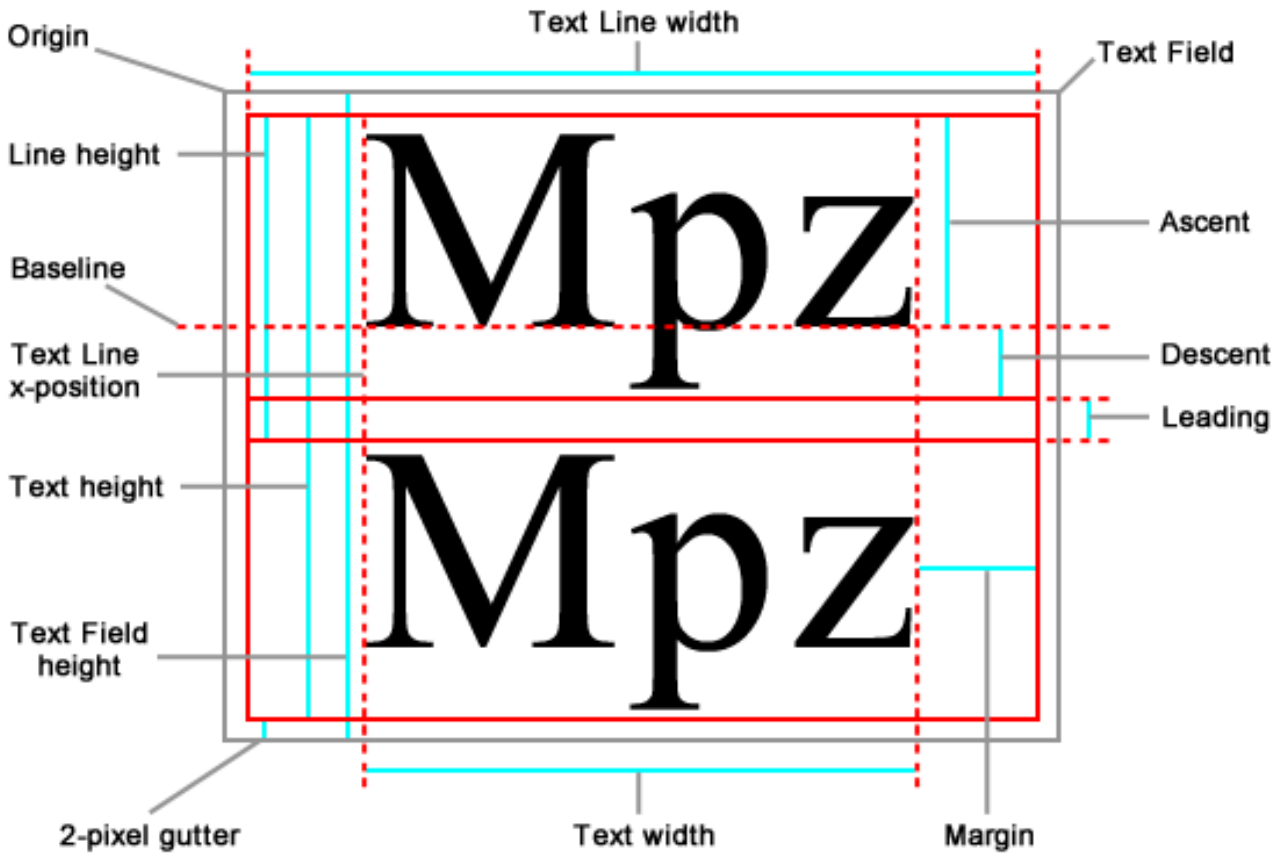
文本行之間的垂直距離值。

`width : Number`

以圖元爲單位描述被選擇文本行的寬度值（不需要全部文本空間）。

`x : Number`

以圖元爲單位描述首字元的左側位置。



參數

`lineIndex:Number` -以零爲基準的文本行索引值（如，第一行文本行爲 0，第二行文本行爲 1，依次類推）。

返回值

Number – 一個物件的格式資訊。

getLineOffset () 方法

```
public function getLineOffset(lineIndex:Number): Number
```

Scaleform 版本： 2.0.37

用 `lineIndex` 參量返回段落中第一個字元的索引。

參數

`lineIndex:Number` - 以零爲基準的文本行索引值（如，首行爲 0，第二行爲 1，依次類推）。

返回值

Number -以零爲基準的文本行中第一個字元索引。

getLineText () 方法

```
public function getLineText(lineIndex:Number): String
```

Scaleform 版本： 2.0.43

用 `lineIndex` 參量返回文本行中的文本索引。

參數

`lineIndex:Number` - 以零爲基準的文本行索引值（如，首行爲 0，第二行爲 1，依次類推）。

返回值

String – 特定行中包含的文本串。

hitTestDisable 屬性

```
hitTestDisable:Boolean  [讀-寫]
```

Scaleform 版本： 1.2.32

設置爲 true 時，當産生點擊動作時 MovieClip.hitTest 函數將忽略該文本域。並且，所有滑鼠事件對該文本域都無效。

預設值爲 false。

noTranslate 屬性

noTranslate:Boolean [讀-寫]

Scaleform 版本 1.2.34

設置爲 true 時，禁止文本域的 GFxTranslator::Translate 函數調用。

預設值爲 false。

numLines 屬性

numLines:Number [讀-寫]

Scaleform 版本 2.0.43

描述多行文本域中的文本行數量。如果 wordWrap 屬性設置爲 true，文本覆蓋時文本行數量將遞增。

topmostLevel 屬性

topmostLevel:Boolean [讀-寫]

Scaleform 版本： 2.1.50

若屬性設置爲 true，文字將顯示在其他文字的上層，忽略其層疊深度。該屬性在實現自定義滑鼠和彈出式視窗時需要在物件前面顯示時有用。預設值爲 false。

在將幾個特性標識為 “topmostLevel” 時,繪圖命令如下所示:

- 對於 Scaleform 3.0.71 而言,將特性標識為 “topmostLevel” 的繪圖命令取決於將這一屬性設置為“真”的命令(因此,首先被標識為“置頂”的特性將首先繪出);
- 從 Scaleform 3.0.72 開始,繪圖命令都是相同的,因為它不再標識“置頂”特性,即將物件 A 繪製在物件 B 下麵,置頂後物件 A 仍然位於物件 B 下方,而無論使用何種命令將 “topmostLevel” 的屬性設置為“真”。

注釋:一旦某一屬性被標識為 “topmostLevel”,swapDepth ActionScript 功能將不會對這一特性產生任何影響。

可參考：

```
MovieClip.topmostLevel
Button.topmostLevel
```

focusGroupMask 屬性

```
focusGroupMask : Number
```

Scaleform 版本： 3.3.84

此屬性將一個位元遮罩設置為一個舞臺人物及其全部 (ALL) 子項。此位元遮罩將焦點組所有權指定給該角色,意思是只有位元遮罩中表明的控制器才能將焦點移動到角色內並在其中移動焦點。可以使用 setControllerFocusGroup 擴展方法將焦點組與控制器相關聯。

例如,我們假設“按鈕 1”(button1) 只能作為控制器 0 的焦點,而“電影剪輯 2”(movieclip2) 只能作為控制器 0 和 1 的焦點。要實現這一行為,請將焦點組與控制器相關聯：

```
Selection.setControllerFocusGroup(0, 0);
Selection.setControllerFocusGroup(1, 1);

button1.focusGroupMask = 0x1; // bit 0 – focus group 0
movieclip2.focusGroupMask = 0x1 | 0x2 // bits 0 and 1 – focus groups 0 and 1
```

“focusGroupMask” 位元遮罩可以設置為父電影剪輯 (parent movieclip)。這將把遮罩值傳播到所有其子項。

7.2 選擇和 剪貼操作

本節介紹選擇和剪貼操作的擴展屬性。

alwaysShowSelection 屬性

```
alwaysShowSelection:Boolean  [讀-寫]
```

Scaleform 版本 2.1.45

默認情況下，文本域無焦點獲得。Scaleform 不高亮顯示被選擇內容。當該屬性設置爲 true 時，即使文本域沒有獲得焦點，Scaleform 也高亮顯示被選擇文本爲灰色。選擇和未選擇顏色區分將被覆蓋（見“可參考”）。

預設值爲 false。

可參考：

selectionBkgColor
selectionTextColor
inactiveSelectionBkgColor
inactiveSelectionTextColor

copyToClipboard () 方法

```
public function copyToClipboard([richClipboard:Boolean], [startIndex:Number],
[endIndex:Number]):void
```

Scaleform 版本： 2.1.45

複製文本到剪貼板。參數選項如下。

參數

`richClipboard:Boolean` – 若爲 true，則文本及其類型資訊複製到剪貼板。預設值與 `useRichTextClipboard` 屬性一致。

`startIndex:Number` – 複製文本段的起始索引。預設值與 `selectionBeginIndex` 相同。

`endIndex:Number` – 複製文本段的終止索引。預設值與 `selectionEndIndex` 相同。

可參考

useRichTextClipboard
selectionBeginIndex
selectionEndIndex
cutToClipboard()
pasteFromClipboard()

cutToClipboard ()方法

```
public function cutToClipboard([richClipboard:Boolean], [startIndex:Number],
[endIndex:Number]):void
```

Scaleform 版本 2.1.45

複製和刪除文本到剪貼板。參數選項如下。

參數

`richClipboard:Boolean` – 若爲 true，則文本及其類型資訊複製到剪貼板。預設值與 useRichTextClipboard 屬性一致。

`startIndex:Number` – 複製文本段的起始索引。預設值與 `selectionBeginIndex` 相同。

`endIndex:Number` – 複製文本段的終止索引。預設值與 `selectionEndIndex` 相同。

可參考：

`useRichTextClipboard`
`selectionBeginIndex`
`selectionEndIndex`
`copyToClipboard()`
`pasteFromClipboard()`

inactiveSelectionBkgColor 屬性

`inactiveSelectionBkgColor:Number` [讀-寫]

Scaleform 版本 2.1.45

指定被選擇活動文本背景顏色。只在 `alwaysShowSelection` 屬性設置爲 true 時有效。顏色的描述格式爲：0xAARRGGBB ，其中 AA 表示 alpha 透明通道十六進位表示數[0,255]，RR - 紅色十六進位表示數[0,255]，BB - 藍色十六進位表示數[0,255]，GG - 綠色十六進位表示數[0,255]。

注釋，確保 alpha 透明通道設置爲 0，因爲本例中無需繪製任何圖畫（因爲 alpha 設置爲 0 表示完全透明）。因此，這種顏色與 Flash 中經常使用的背景色略有不同。例如，設置該屬性填充深紅色需要使用 0xFFFF0000 值（alpha 和紅色設置爲 0xFF (255)），但是對於常用的“backgroundColor”屬性使用 0xFF0000 值就已足夠。

可參考：

`alwaysShowSelection`
`inactiveSelectionTextColor`
`selectionBkgColor`
`selectionTextColor`

inactiveSelectionTextColor 屬性

```
inactiveSelectionTextColor:Number  [讀-寫]
```

Scaleform 版本： 2.1.45

指定被選擇活動文本顏色。只在 alwaysShowSelection 屬性設置爲 true 時有效。顏色的表示格式爲：0xRRGGBB ，其中 AA 表示 alpha 透明通道十六進位表示數[0,255]，RR - 紅色十六進位表示數[0,255]，BB - 藍色十六進位表示數[0,255]，GG - 綠色十六進位表示數[0,255]。

注釋，確保 alpha 透明通道設置爲 0，因爲本例中無需繪製任何圖畫（因爲 alpha 設置爲 0 表示完全透明）。因此，這種顏色與 Flash 中經常使用的背景色略有不同。例如，設置該屬性填充深紅色需要使用 0xFFFF0000 值（alpha 和紅色設置爲 0xFF (255)），但是對於常用的“textColor”屬性使用 0xFF0000 值就已足夠。

可參考：

```
alwaysShowSelection
inactiveSelectionBkgColor
selectionBkgColor
selectionTextColor
```

noAutoSelection 屬性

```
noAutoSelection:Boolean  [讀-寫]
```

Scaleform 版本： 2.1.45

若設置爲 true，焦點切換到文本域禁止自動選擇。

默認爲 false。

pasteFromClipboard () 方法

```
public function pasteFromClipboard([richClipboard:Boolean], [startIndex:Number],
[endIndex:Number]):void
```

Scaleform 版本： 2.1.45

從剪貼板粘貼文本。參數選項如下。

參數

`richClipboard:Boolean` – 若爲 true，文本類型屬性從剪貼板粘貼。預設值與 useRichTextClipboard 屬性相同。

`startIndex:Number` – 從剪貼板粘貼文本段的起始索引。預設值與 `selectionBeginIndex` 相同。

`endIndex:Number` – 從剪貼板粘貼文本段的終止索引。預設值與 `selectionEndIndex` 相同。

可參考：

`useRichTextClipboard`
`selectionBeginIndex`
`selectionEndIndex`
`copyToClipboard()`
`cutToClipboard()`

selectionBeginIndex 屬性

`selectionBeginIndex:Number  [讀-寫]`

Scaleform 版本： 2.1.45

表示以零爲基準的當前選擇字串中第一個字元索引值。若文本未被選擇，本屬性值即爲 `caretIndex` 的值。本屬性與 `Selection.getBeginIndex()`函數的返回值相同；不同的是 `Selection.getBeginIndex()`函數在文本未獲得焦點時也可使用，此時，`Selection.getBeginIndex()` 的返回值爲-1。

可參考：

`caretIndex`
`selectionEndIndex`
`Selection.getBeginIndex()`

selectionEndIndex 屬性

`selectionEndIndex:Number  [讀-寫]`

Scaleform 版本： 2.1.45

表示以零爲基準的當前選擇字串中最後一個字元索引值。若文本未被選擇，本屬性值即爲 `caretIndex` 的值。本屬性與 `Selection.getEndIndex()`函數的返回值相同；不同的是 Selection.getEndIndex()函數在文本未獲得焦點時也可使用，此時，`Selection.getEndIndex()`的返回值爲-1。

可參考：

```
caretIndex
selectionBeginIndex
Selection.getEndIndex()
```

selectionBkgColor 屬性

```
selectionBkgColor:Number  [讀-寫]
```

Scaleform 版本： 2.1.45

描述當前被選擇文本的背景色。顏色的表示格式爲：0xRRGGBB ，其中 AA 表示 alpha 透明通道十六進位表示數[0,255]，RR - 紅色十六進位表示數[0,255]，BB - 藍色十六進位表示數[0,255]，GG - 綠色十六進位表示數[0,255]。

注釋，確保 alpha 透明通道設置爲 0，因爲本例中無需繪製任何圖畫（因爲 alpha 設置爲 0 表示完全透明）。因此，這種顏色與 Flash 中經常使用的背景色略有不同。例如，設置該屬性填充深紅色需要使用 0xFFFF0000 值（alpha 和紅色設置爲 0xFF (255)），但是對於常用的 `BkgColor` 屬性使用 0xFF0000 值就已足夠。

可參考：

```
inactiveSelectionTextColor
inactiveSelectionBkgColor
selectionTextColor
```

selectionTextColor 屬性

```
selectionTextColor:Number  [讀-寫]
```

Scaleform 版本： 2.1.45

描述當前被選擇文本的顏色。顏色的表示格式爲：0xRRGGBB ，其中 AA 表示 alpha 透明通道十六進位表示數[0,255]，RR- 紅色十六進位表示數[0,255]，BB - 藍色十六進位表示數[0,255]，GG - 綠色十六進位表示數[0,255]。

注釋，確保 alpha 透明通道設置爲 0，因爲本例中無需繪製任何圖畫（因爲 alpha 設置爲 0 表示完全透明）。因此，這種顏色與 Flash 中經常使用的背景色略有不同。例如，設置該屬性填充深紅色需要使用 0xFFFF0000 值（alpha 和紅色設置爲 0xFF (255)），但是對於常用的 `textColor` 屬性使用 0xFF0000 值就已足夠。

可參考：

```
inactiveSelectionBkgColor
inactiveSelectionBkgColor
selectionBkgColor
```

useRichTextClipboard 屬性

```
useRichTextClipboard:Boolean  [讀-寫]
```

Scaleform 版本： 2.1.45

指定是否在文本複製和粘貼操作時帶上文本格式資訊。當設置爲 `true` 時，Scaleform 在複製和粘貼文本時同時也複製和粘貼文本格式（如對齊、粗體和斜體等）。要實現此功能，複製和粘貼過程中的原始和目標文本域的屬性 `useRichTextClipboard` 必須設置爲 `true`。

預設值爲 `false`。

7.3 文本大小和對齊

本節介紹文本大小和對齊控制。除了標準的對齊格式（左對齊、右對齊、居中），Scaleform 提供了垂直對齊，也就是垂直大小自適應功能。

fontScaleFactor 屬性

```
fontScaleFactor:Number  [讀-寫]
```

Scaleform 版本： 2.0.43

指定文本域中所有字體的字形縮放。所有字體大小可以通過改變 `fontScaleFactor` 值而改變。

預設值爲 1.0。

textAutoSize 屬性

```
textAutoSize:String  [讀-寫]
```

Scaleform 版本： 2.0.43

textAutoSize 能夠自動恢復文本字體大小。本屬性的有效值爲“none”、“shrink”和“fit”。如果該模式設置有效（“shrink”或“fit”）同時文本不適應于文本域，則文本大小按比例縮放到與文本域匹配爲止，無需滾動操作。如果文本過小（字體大小爲 5pt 左右）則默認的操作邏輯將停止字體縮放動作。設置 textAutoSize 爲“fit”模式時，字體大小增大到填充滿整個文本空間位置。設置爲“shrink”模式時當字體大小超出文本空間範圍時將減小字體的大小。

預設值爲“none”。

verticalAlign 屬性

`verticalAlign:String [讀-寫]`

Scaleform 版本： 2.0.43

設置文本框中文字的垂直對齊。該屬性的有效值爲：“none”（或“top”與“none”相同），“bottom”、“center”。若屬性設置爲“center”文本在文本域內居中，若設置爲“bottom”，則在文本域中底部對齊。

預設值爲“none”。

可參考：

`verticalAutoSize`
`TextField.align`

verticalAutoSize 屬性

verticalAutoSize:String [讀-寫]

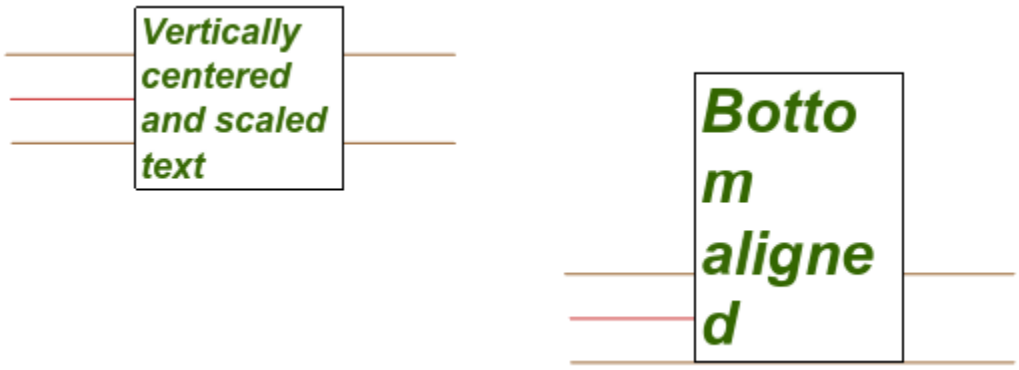
Scaleform 版本： 2.0.43

文本域的自動垂直大小控制和對齊。有效值爲“none”（默認）、“top”、“bottom”、“center”。如果 verticalAutoSize 設置爲“none”（預設值）將不會改變大小。

若 verticalAutoSize 設置爲“top”，文本域中特性與 TextField.autoSize 模式下 wordWrap 設置爲 true 時相同（只改變文本域底部邊界，不改變右邊界）。

若 verticalAutoSize 設置爲“center”，文本域將上下均勻改變其大小。定位點位元於文本域上下邊的中點（參考下圖）。

若 verticalAutoSize 設置爲“bottom”，文本域將改變頂部邊界。定位點位元於文本域底部邊界（參考下圖）。



預設值爲“none”。

可參考：

verticalAlign
TextField.autoSize

7.4 HTML 擴展

本節介紹 Scaleform 中 HTML 的擴展功能。

<FONT> 標簽， ALPHA 特性

ALPHA="#xx"

設置文本 alpha 透明度，範圍爲 00 到 FF（0-255），0 代表全透明，255 代表全不透明。與 COLOR 屬性相對應，ALPHA 屬性用來設置 HTML 中的半透明文本。

<IMG> 標簽

當前， Scaleform 2.0/2.1 中的<IMG> 標簽用相應的鏈結導入圖像到庫。Scaleform 不支援從 Web 或者文件等外部資料源導入圖像(不支援"http://", "file://" 及其他協定）。 上面描述如何導入到圖像並指派鏈結。

以下爲一個 HTML 使用<IMG>標簽導入“myImage”圖像的例子：

```
t.htmlText = "<p align='right'>abra<img src='myImage' width='20' height='30' align='baseline' vspace='-10'>bed232</p>";
```

當前支援的 IMG 標簽屬性：

`src` –為庫中的圖像符號指定連結識別字。目前僅支援圖像。此屬性為必選;所有其他屬性均為可選。目前還不支援外來檔(JPEG、GIF、PNG 和 SWF 檔)。這裡可以使用使用者協定 “img://”,取代內嵌影像連結識別字;在此情況下,將調用使用者定義的虛擬方法 GFx::ImageCreator::LoadImage。

`width` – 插入圖像的寬度資訊，以圖元表示。

`height` – 插入圖像的高度資訊，以圖元表示。

`align` – 指定文本域中嵌入圖像的水平對齊屬性。當前支援的唯一屬性爲“baseline”。

`vspace` – “baseline”對齊方式以圖元爲單位指定相對於基準線的偏移量：正值表示圖像位於基線之上，負數則降低圖像位置。

不支援屬性：

`id` – 指定包含插入圖像文件的視頻剪輯實例名字（由 Flash Player 創建），圖像文件來自 SWF 文件或者視頻剪輯。

`align` – “left”和“right”。指定文本域中插入圖像的垂直對齊方式。

`hspace` - “left”和“right”。指定圍繞圖像無文本區域的垂直空間。

注釋，如果圖像從另一個 SWF 文件中導入，則圖像在 Flash 文件中應該用 IMG 標簽明確指定。如果圖像導入但未在 Flash 中使用，則在最終的 SWF 文件中將被丟棄。爲防止圖像丟棄，可創建一個視頻剪輯並用 IMG 標簽指定所有導入圖像。不要遺忘導入視頻剪輯，如果不明確使用並沒有導出則 Flash 同樣會在 SWF 文件中丟棄圖像。

7.5 陰影效果控制

本節介紹不同陰影效果的控制（blur，knockout 等）。

blurX, blurY 特性

```
blurX:Number  [讀-寫]
blurY:Number  [讀-寫]
```

Scaleform 版本： 2.0.41

獲得或者設置文本 blur 效果半徑。有效值範圍爲 0 到 15.9(浮點數)。

預設值爲 0。

可參考：

```
blurStrength
```

blurStrength 特性

```
blurStrength:Number  [讀-寫]
```

Scaleform 版本： 2.0.41

獲得或者設置文本 blur 強度。有效值範圍爲 0 到 15.9（浮點數）。

預設值爲 0。

可參考：

```
blurX
blurY
```

shadowAlpha 特性

```
shadowAlpha:Number  [讀-寫]
```

Scaleform 版本： 2.0.41

控制色彩陰影的 alpha 透明值。有效值範圍爲 0.00 到 1.00，例如 0.25 設置 25%透明度。

可參考：

```
shadowColor
```

shadowAngle 特性

```
shadowAngle:Number  [讀-寫]
```

Scaleform 版本： 2.0.41

控制陰影角度，與 DropShadowFilter 類似。有效值範圍爲 0 到 360˚ (浮點數)。

預設值爲 45。

可參考：

```
DropShadowFilter
```

shadowBlurX, shadowBlurY 特性

```
shadowBlurX:Number  [讀-寫]
shadowBlurY:Number  [讀-寫]
```

Scaleform 版本： 2.0.41

控制文本 blur 效果半徑。有效值範圍爲 0 到 15.9(浮點數)。

預設值爲 0。

可參考：

```
shadowStrength
```

shadowColor 特性

shadowColor:Number [讀-寫]

Scaleform 版本： 1.1.31

指定陰影顏色（詳情請參考 shadowStyle）。顏色的表示格式爲：0xRRGGBB ，其中 RR - 紅色十六進位表示數[0..255]，BB - 藍色十六進位表示數[0..255]，GG - 綠色十六進位表示數[0..255]。

可參考：
shadowStyle

shadowDistance 特性

shadowDistance:Number [讀-寫]

Scaleform 版本： 2.0.41

以圖元爲單位元控制陰影偏移距離，與 DropShadowFilter 相同。

可參考：
DropShadowFilter

shadowHideObject 特性

shadowHideObject:Boolean [讀-寫]

Scaleform 版本： 2.0.41

指示文本是否隱藏。值爲 true 表示文本本身不現實，只顯示其陰影。預設值爲 false（顯示文本本身）。

shadowKnockOut 特性

shadowKnockOut:Boolean [讀-寫]

Scaleform 版本： 2.0.41

空心效果控制和（true），該效果令物件透明填充，可以看到其文檔背景的顏色。預設值爲 false（無空心效果）。

shadowQuality 特性

`shadowQuality:Number  [讀-寫]`

Scaleform 版本： 2.0.41

控制陰影或者濾光的品質。與 Flash 不同，唯一的設置值爲 1 和 2。值爲 1 表示低品質，2 或者更大值表示高品質。

shadowStrength 特性

`shadowStrength:Number  [讀-寫]`

Scaleform 版本： 2.0.41

控制陰影模糊強度。有效值範圍爲 0-15.9（浮點數）。

預設值爲 0。

可參考：

- `shadowBlurX`
- `shadowBlurY`

shadowStyle 特性

`shadowStyle:String  [讀-寫]`

Scaleform 版本： 1.1.31

與 `shadowColor` 相結合，控制文本域的陰影渲染。`shadowStyle` 格式字串描述了文本層的設置，與應用物件相匹配。例如：

```
field.shadowStyle = "s{0,-1.5}{-1.4,1.2}{1.4,1.2}t{0,0}";
field.shadowColor = 0xff0000;
```

字元“s”界定了陰影圖層採用 `shadowColor` 值的顏色表示，而字元“t”界定了文本層用標準的 `TextField.textColor` 顏色值表示。這些特性在每個文本層將會被使用並渲染；但相同的轉換應用到文本，每個單元映射到一個圖元。如果“t”界定並且設置爲“t（0，0），則忽略其設置效果。

請注意當前的實現當中使用陰影層，由於多次渲染將產生額外的棱角和粗糙感。強烈建議測量性能結果並盡可能得限制陰影的使用。

可參考：

```
shadowColor
TextField.textColor
```

7.6 圖像替換

本節介紹了圖像替換的控制方法。

setImageSubstitutions() 方法

```
public setImageSubstitutions(substInfoArr:Array) : void
public setImageSubstitutions(substInfo:Object) : void
public setImageSubstitutions(null) : void
```

Scaleform 版本： 2.0.38

設置文本域圖像替換鏈結。
只有在圖像插入到 SWF 時字串鏈結替換才有效；這些圖像還需要匹配鏈結以輸出名字。插入圖像到 SWF 時需要用到：

1. 導入點陣圖圖像到庫。
2. 右鍵點擊（Windows 系統）或者控制鍵點擊（Macintosh 系統）位於庫中的圖像，從功能表目錄中選擇鏈結項。
3. 選擇 ActionScript 輸出，在第一幀中輸出並在標識文本框中輸入其名字（例如，myImage）。
4. 點擊 OK 確定鏈結符。

在圖像導入鏈結符分配完成後，需要創建一個 BitmapData 實例。下面爲該實例的 ActionScript 代碼示例：

```
import flash.display.BitmapData;
var imageBmp:BitmapData = BitmapData.loadBitmap("myImage");
```

注意：不要遺忘導入聲明(導入 `import flash.display.BitmapData`;或使用完整名稱 - `flash.display.BitmapData`)，不然導致結果“未定義”！

如果需要使用多個圖像做替換，需要爲每個圖像重復這些步驟，給出不同的鏈結 ID。

單個替換物件設置如下：

`subString:String`
　　定義替換圖像的子鏈結；這個爲必要項。最大子鏈長度爲 15 個字元。

`image : BitmapData`
　　指定圖像；必要項。

`width : Number`
　　定義螢幕中的圖像顯示寬度，以圖元爲單位。可選項。

`height : Number`
　　定義螢幕中的圖像顯示高度，以圖元爲單位。可選項。

`baseLineY : Number`
　　定義圖像基準線的 Y 座標偏移量，以原始圖像的圖元爲單位（w/o 轉換）。可選項。默認情況下，該值與圖像高度一致，因此，圖像底部正好位於基準線上。

`id : String`
　　指定替換 ID 作爲“updateImageSubstitution ”調用的的第一個參數。可選項。

“`substInfoArr`”應該爲這些物件的序列，如同“`substInfo`”應該爲物件的實例一樣。版本 `setImageSubstitutions(substInfo:Object)`只支援單個替換，而版本 `setImageSubstitutions(substInfoArr:Array)`可設置多個。

注意，每個 `setImageSubstitutions` 的調用增加內部列表的替換。清楚這些元素需要調用 `setImageSubstitutions(null)`。

`setImageSubstitutions` 函數調用後，在 ActionScript 中必須保持一個到替換序列或者單個描述物件的索引；無論如何，需要保持這個索引，由於無法將替換序列恢復到文本域中，在 ActionScript 的其他地方需要引用則需要這個索引。

參數

`substInfoArr:Array` –	描述符物件替換序列（見上）。
`substInfo:Object` –	單個描述符物件替換（見上）。
`null` –	清除所有替換。

可參考：
　　`updateImageSubstitution()`

示例：

```
var b1 = BitmapData.loadBitmap("smile1");
var b2 = BitmapData.loadBitmap("smile2");
var b3 = BitmapData.loadBitmap("smile3");
var a = new Array;
a[0] = { subString:"=)", image:b1, baseLineY:35, width:20, height:20, id:"sm=)" };
a[1] = { subString:":-)", image:b2, baseLineY:20, id:"sm:-)" };
a[2] = { subString:":\\", image:b3, baseLineY:35, height:100 };
a[3] = { subString:":-\\", image:b1 };
t.setImageSubstitutions(a);
```

只要文本域中包含子鏈“=）”，當並不應用，該子鏈將被名爲“smile1”圖像鏈結符所替換。

updateImageSubstitution() 方法

```
public updateImageSubstitution(id:String, image:BitmapData) : void
```

Scaleform 版本： 2.0.38

替換或刪除原先由 `setImageSubstitutions` 函數爲文本替換創建的圖像。

參數

`id:String` –	替換 ID 號，與 `setImageSubstitutions` 函數調用使用的描述符物件 ID 成員相同。
`image:BitmapData` –	指定新圖像；若爲 null 則完全刪除替換。

可參考：

`setImageSubstitutions()`

示例：

```
t.updateImageSubstitution("sm=)", b3);
```

以下爲插入圖像的動畫示例。在 onEnterFrame 控制碼或使用 setInterval 可作更新操作。注意，`updateImageSubstitution` 調用時不會發生文本重新格式化；因此，新圖像的大小應該與原先圖像保持一致。

```
var phase = 0;
var b1a = BitmapData.loadBitmap("smile1a");
var b2a = BitmapData.loadBitmap("smile2a");

onEnterFrame = function()
{
```

```
if (phase % 10 == 0)
{
   if (phase % 20 == 0)
   {
      _root.t.updateImageSubstitution("sm=)", b1);
      _root.t.updateImageSubstitution("sm:-)", b2);
   }
   else
   {
      _root.t.updateImageSubstitution("sm=)", b1a);
      _root.t.updateImageSubstitution("sm:-)",b2a);
   }
}
++phase;}
```

7.7 IME 支援

本節介紹擴展 IME 支援。

disableIME 特性

```
disableIME:Boolean  [讀-寫]
```

Scaleform 版本： 2.1.50

若設爲 true，之前 IME 將在當前文本域有效。

預設值爲 false。

getIMECompositionStringStyle() 方法

```
public function getIMECompositionStringStyle(categoryName:String): Object
```

Scaleform 版本： 2.1.50

返回 IME 顏色設置種類中的顏色設置描述符物件。參閱 `setIMECompositionStringStyle` 獲得 cateforyName 及其描述符物件的詳細描述資訊。

參數

categoryName:String – IME 顏色設置類別（查看 `setIMECompositionStringStyle` ）。

返回值

物件- 顏色設置目錄描述符（查看 `setIMECompositionStringStyle` ）。

可參考：

`setIMECompositionStringStyle`
`disableIME`

setIMECompositionStringStyle() 方法

```
public function setIMECompositionStringStyle(categoryName:String,
                                    styleDescriptor:Object): Number
```

Scaleform 版本： 2.1.50

設置 IME 顏色設置類別類型。categoryName 參數可有如下值：

"compositionSegment" – 設置整個字元的顏色屬性；
"clauseSegment" – 設置子段顏色屬性；
"convertedSegment" – 設置轉換文本段顏色屬性；
"phraseLengthAdj" – 設置慣用語長度調整顏色屬性；
"lowConfSegment" – 設置有錯字元顏色屬性。

styleDescriptor 爲一個物件實例，有以下選項：

textColor : Number
文本顏色。

backgroundColor : Number
背景色。

underlineColor : Number
下劃線顏色。

underlineStyle : String
下劃線類型。有效值：
“single”
“thick”
“dotted”
“ditheredSingle”
“ditheredThick”

所有顏色的表示格式爲：0xRRGGBB ，其中 RR-紅色十六進位表示數[0..255]，BB-藍色十六進位表示數[0..255]，GG-綠色十六進位表示數[0..255]。

參數

categoryName:String – IME 顏色設置類別（見描述資訊）。
styleDescriptor:Object – 顏色設置描述符物件（見描述資訊）。

可參考：

getIMECompositionStringStyle
disableIME

8 擴展 TextFormat 類

TextFormat 類表示字元格式資訊，如顏色、字體大小、下劃線等。

alpha 特性

`alpha:Number  [讀-寫]`

Scaleform 版本： 2.0.41

按照比例控制文本 alpha 透明度。有效值範圍爲 0-100%。標準 Flash `TextFormat.color` 不允許設置 alpha 透明度，因此，這個擴展功能不允許用來設置文本爲部分或者完全透明。

可參考：

- `TextFormat`
- `HTML <FONT ALPHA='xx'>`

9 擴展 Stage 類

除了支援標準 Stage 類特性，Scaleform 引入了擴展功能增強了場景維度跟蹤。擴展屬性包括“visibleRect”，“safeRect”和“originalRect”，同時還提供了“onResize”額外參數，用來表示當前可視幀形狀。詳情見下。

注意，在 ActionScript 1.0 中可以直接引用擴展屬性，如 Stage.visibleRect。而 ActionScript 2.0 中不能直接引用。爲減輕 ActionScript 2.0 編譯器負擔，應該選用其他訪問語法如：Stage["visibleRect"]。

visibleRect 特性

visibleRect:flash.geom.Rectangle [讀-寫]

Scaleform 版本： 2.2.56

本特性包含了當前可視矩形。該矩形會根據縱橫比、縮放模式、對齊和 Viewport 縮放等參數的改變而改變，並且需要知道當前可視物件。該矩形具有負 topLeft 位置值。

可參考：

- safeRect
- originalRect

safeRect 特性

safeRect:flash.geom.Rectangle [讀-寫]

Scaleform 版本： 2.2.56

本特性包含了當前安全矩形設置資訊。應該由用戶使用 Scaleform::Movie::SetSafeRect 來設置。若無設置，則與“visibleRect”包含相同矩形。

可參考：

- visibleRect
- originalRect

originalRect 特性

```
originalRect:flash.geom.Rectangle  [讀-寫]
```

Scaleform 版本： 2.2.56

該特性總是包含原始 SWF 解析度的 SWF 原始矩形。因此，若 Flash Studio 中 “Document properties” 解析度設置爲（600pxx400px），則該矩形包含值爲{ 0, 0, {600, 400} }。

可參考：

`visibleRect`
`safeRect`

onResize 事件控制碼

```
onResize = function([visibleRect:flash.geom.Rectangle]) {}
```

Scaleform 版本： 2.2.56

onResize 事件控制碼擁有一個擴展參數。該參數描述了當前可視矩形，該矩形與 `Stage.visibleRect` 返回的擴展特性相同。

可參考：

`visibleRect`

translateToScreen() 方法

```
public function translateToScreen(pt:Object): Point
```

Scaleform 版本： 3.3.84

此方法返回 Stage 坐標系中的映射到螢幕坐標系的一個點。

參數

`pt:Object` –具有代表要轉換的點的座標的 x 和 y 成員的一個 Object。可以使用 flash.geom.Point,而非通用 Object。

返回

點 – 映射到螢幕坐標系的輸入點。

10 陣列類擴展

指定分類排序

Flash 版本中 Array.sort 和 Array.sortOn 根據 Unicode 值來進行分類排序。但是，這樣排序結果經常不對，特別是用在法語、西班牙語、德語等。這些語言中使用 Unicode 分類排序，所有的字串字幕 'z' 的後面都跟隨音標符號（例如，元音符、重音符、強調）。因此，在 Flash 中排序陣列['á', 'z', 'a'] 返回 ['a', 'z', 'á'] 。但是，對於法語則返回爲['a', 'á', 'z']。

爲解決這個問題，Scaleform 引入了另外一個選項 Array.sort/sortOn 方法：

Array.LOCALE:

```
var arr = ['á', 'z', 'a];
var newArr1 = arr.sort(); // 返回 ['a', 'z', 'á']
var newArrLoc = arr.sort(Array.LOCALE); // 返回 ['a', 'á', 'z']
var newArrRev = arr.sort(Array.LOCALE | Array.DESCENDING); // ['z', 'á', 'a']
```

也支援大小寫敏感局部指定分類排序：

```
var arr = ['Á', 'z', 'a'];
var a = arr.sort(Array.LOCALE | Array.CASEINSENSITIVE); // 返回
['a', 'Á', 'z']
```

注釋，本函數可能在特定平臺上不能正確工作，有些平臺內核不支援該函數。這種情況下 Array.LOCALE 分類即被當作按照普通 Unicode 值分類。

Scaleform 版本： 3.0.65

參考：

String.localeCompare

11 String 類擴展

localeCompare()方法

本函數比較兩個或者更多字串類型並以數位值的形式返回比較結果。本方法以局部指定方式來作比較。如果字串相同，返回值爲 0，如果原字串中部分和參數中指定的比較字串相同則返回一個負值，如果原字串與參數中指定的比較字串中部分內容相同，則返回一個正值。

見“陣列類擴展”小節獲得更多關於局部指定字串操作資訊。
注釋，本函數可能在特定平臺上不能正確工作，可能有些平臺內核不支援該函數。這種情況下，本函數即被當作 Unicode 值的常規字串比較函數。

```
public localeCompare(other:String [, caseInsensitive:Boolean]) : Number
```

Scaleform 版本： 3.0.65

參數

`other:String` –	比較字串 。
`caseInsensitive:Boolean` –	一個可選參數可被用來做忽略大小字母比較。如果參數未被指定，則默認爲大小寫敏感比較。

參考：

`Array.sort`

12 視頻類擴展

在標準的 Flash 中只有一個視頻物件可以每次從 NetStream 物件收到資料。Scaleform 取消了這項限制並允許多個視頻物件附加到 NetStream 物件，可以接受相同視頻資料。

clipRect 屬性

`clipRect: flash.geom.Rectangle  [讀-寫]`

Scaleform 版本： 3.0.70

該屬性允許視頻物件可以只顯示原始視頻幀的一部分（區域）。

例如：

```
video.clipRect = new flash.geom.Rectangle(10, 20, 100, 100);
```

13 3Di Extensions

Scaleform 3Di 是使用一組內置到 Scaleform 中的基本 AS2 擴展實現的。為 3Di 增加了下列 ActionScript 擴展：

Scaleform 版本： 3.2.82

_z

設置物件的 Z 座標(深度),預設值為 0。

_zscale

將 Z 向的物件伸縮設置一個百分比,預設值為 100。

_xrotation

設置物件圍繞 X(橫向)軸旋轉,預設值為 0。

_yrotation

設置物件圍繞 Y(縱向)軸旋轉,預設值為 0。

_matrix3d

使用一組 16 個浮點數(4 x 4 矩陣)設置物件的完整 3D 轉換。如果此值設置為 NULL,就會刪除所有 3D 轉換,而且物件將會變為 2D。

perspfov

在物件上設置透視視野 (Field Of View) 角度,有效角度必須 > 0 度和< 180 度。 180=""></ 180 度。>如果不設置,物件就繼承根 FOV 角,該角預設為 55 度。

使用這些新擴展需要在 ActionScript 中把全域變數 gfxExtensions 設置為 True(真)。

14 全局擴展

noInvisibleAdvance 特性

```
noInvisibleAdvance : Boolean
```

Scaleform 版本：2.1.52

若設置爲 `true`，本特性關閉所有隱藏視頻剪輯的前進動作。這將提高擁有大量隱藏視頻剪輯的 SWF 性能。注意，Flash 中隱藏視頻剪輯也有前進動作（將仍然執行時間線動畫，調用各幀的 ActionScript 腳本等）。因此，將此特性設置爲“`true`”，可以在 Scaleform 和 Flash 中産生不同的效果。

可參考：

`MovieClip.noAdvance`

Imecommand() 函數

```
imecommand(command:String, parameters:String) : Void
```

Scaleform 版本： 2.1.50

本函數類似於 fscommand，但只用來與 Scaleform IME 執行器交互資料。在 Scaleform 內部使用。

getIMECandidateListStyle() 函數

```
public function getIMECompositionStringStyle(categoryName:String): Object
```

Scaleform 版本： 2.1.50

返回 IME 選擇列表的顔色設置描述符。查看 `setIMECandidateListStyle` 獲得更多的描述符物件資訊。

返回

Object – 文字候選列表顔色設置描述器。

可參考：

`setIMECandidateListStyle`
`TextField.disableIME`

setIMECandidateListStyle() 函數

```
public function setIMECandidateListStyle(styleDescriptor:Object)
```

Scaleform 版本： 2.1.50

設置候選列表類型作爲物件傳遞參數。

styleDescriptor 爲一個物件實例，有以下選項：

```
      styleObject.textColor                    = 0x5EADFF;
      styleObject.selectedTextColor            = 0xFFFFFF;
      styleObject.fontSize                     = 20;
      styleObject.backgroundColor              = 0x001430;
      styleObject.selectedTextBackgroundColor  = 0x2C5A99;
      styleObject.indexBackgroundColor         = 0x12325A;
      styleObject.selectedIndexBackgroundColor = 0x2C5A99;
      styleObject.readingWindowTextColor       = 0xFFFFFF;
      styleObject.readingWindowBackgroundColor = 0x001430;
      styleObject.readingWindowFontSize        = 20;
```

textColor : Number
      文本顏色。

selectedTextColor : Number
      已選擇文本顏色。

fontSize : Number
      行中字體大小。

backgroundColor : Number
      未選擇行中文本的背景顏色。

seletedTextBackgroundColor : Number
      選擇行中文本顏色。

indexBackgroundColor : Number
      未選擇行中指標背景顏色。

`selectedIndexBackgroundColor : Number`
　　選擇行中指標背景顏色。

`readingWindowTextColor : Number`
　　閱覽窗中文本顏色。

`readingWindowBackgroundColor : Number`
　　閱覽窗中背景顏色。

`readingWindowFontSize : Number`
　　閱覽窗中文本字體大小。

所有顏色的描述格式爲：0xRRGGBB ，其中 RR - 紅色十六進位表示數[0,255]，BB - 藍色十六進位表示數[0,255]，GG - 綠色十六進位表示數[0,255]。

可參考：
　　`getIMECandidateListStyle`
　　`TextField.disableIME`

15 標準方法和事件處理常式擴展

Key 類

```
 Key.getCode(controllerIdx:Number)
Key.getAscii(controllerIdx:Number)
Key.isDown(controllerIdx:Number)
Key.isToggled(controllerIdx:Number)
```

Key 类方法取一个用于键盘/控制器的可选参数。如果没有指定，则 `controllerIdx` 将默认为 0。

```
Key.onKeyDown(controllerIdx:Number)
```

Key 侦听器方法 onKeyDown 可接收一个用于生成该事件的键盘/控制器的额外参数。

Selection 類

```
Selection.setFocus(ch:Object, controllerIdx:Number)
Selection.getFocus(controllerIdx:Number)
Selection.getBeginIndex(controllerIdx:Number)
Selection.getEndIndex(controllerIdx:Number)
Selection.setSelection(start:Number, end:Number, controllerIdx:Number)
Selection.getCaretIndex(controllerIdx:Number)
```

Selection 類方法取一個用於鍵盤/控制器的可選參數。如果沒有指定,則 controllerIdx 將預設為 0。

```
Selection.onSetFocus(old:Object, new:Object, controllerIdx:Number)
```

Selection 偵聽器方法 onSetFocus 可接收用於生成該事件的鍵盤/控制器的一個額外的參數。

MovieClip/Button/TextField

```
MovieClip.onSetFocus(old:Object, controllerIdx:Number)
MovieClip.onKillFocus(new:Object, controllerIdx:Number)
Button.onSetFocus(old:Object, controllerIdx:Number)
Button.onKillFocus(new:Object, controllerIdx:Number)
TextField.onSetFocus(old:Object, controllerIdx:Number)
TextField.onKillFocus(new:Object, controllerIdx:Number)
TextField.onChanged(controllerIdx:Number)
```

這些 MovieClip/Button/TextField 偵聽器方法可接收用於生成該事件的鍵盤/控制器的一個額外的參數。

System.capabilities

`System.capabilities.numControllers` –返回系統中檢測到的控制器的數量。