Autodesk®
Scaleform®

DrawText API

이 문서는 Scaleform 4.2 에서 사용 가능한 DrawText API 에 대하여 설명한다. 이 API 는 GFx::Movie 나 액션스크립트 샌드박스 외부에서도 C++기반 형태로 문자 렌더링과 포맷팅에 사용 가능하다.

집필: Artem Bolgar

버전: 2.0
최종편집일: 2011 년 4 월 22 일

Autodesk®
GAMEWARE

Copyright Notice

Autodesk® Scaleform® 4.2

Disclaimer

THIS PUBLICATION AND THE INFORMATION CONTAINED HEREIN IS MADE AVAILABLE BY AUTODESK, INC. "AS IS." AUTODESK, INC. DISCLAIMS ALL WARRANTIES, EITHER EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE REGARDING THESE MATERIALS.

연락처:

문서	DrawText API
주소	Autodesk Scaleform Corporation
	6305 Ivy Lane, Suite 310
	Greenbelt, MD 20770, USA
웹사이트	www.scaleform.com
이메일	info@scaleform.com
직통전화	(301) 446-3200
팩스	(301) 446-3199

목차

1 들어가기

Scaleform SDK 는 파워풀한 폰트 렌더링과 텍스트 포맷팅 엔진을 탑재하고 있으며, 이 기능은 주로 플래시 기반의 UI 에서 문자를 출력하기 위해서 사용된다. Scaleform 에서 텍스트 렌더링은 즉시 레스터화, 혹은 스크린-픽셀에 정렬된 글리프(glyph)의 캐싱, 그리고 HTML 태그로 장식된 텍스트를 시스템 폰트나 SWF/GFX 에 임베드 된 데이터로 렌더링 하는 등의 진보된 기능을 지원한다.

대부분의 경우 Scaleform 텍스트 렌더링 시스템은 디자이너가 플래시 스튜디오에서 작업한 플래시 파일을 플레이 하면 자동적으로 사용된다. 여기서 추가해서 액션스크립트(AS)에서 TextField 와 TextFormat 클래스를 통해 외부에서 접근 가능하다. 따라서 이들 API 를 사용하면 C++ API 를 사용할 필요성이 현저하게 줄어들게 된다.

하지만, 때로는 액션스크립트를 통한 텍스트 렌더링이 불편할 수도 있고, 이와 관련된 오버헤드가 부담스러울 수도 있다. 이동중인 아바타의 상단에 이름 빌보드를 렌더링 하는 등의 경우에는 완벽한 플래시 UI 를 지원하지 않는 한 C++을 사용하는 것이 훨씬 효율적일 것이다. 게다가 Scaleform 기반으로 3D 엔진에서 HUD 를 사용한 메뉴 시스템을 통합했다면 동일한 폰트 시스템을 사용하는 것이 필요하다.

Scaleform3 이상 상위버전에서는 C++용 DrawText API 를 탑재하고 있어서 무비뷰 샌드박스 외부에서 Scaleform 폰트 렌더링과 텍스트 포매팅 엔진에 접근할 수 있다. 새로운 텍스트 API 는 여전히 Scaleform 플레이어의 Render::Renderer 인터페이스를 사용하지만, GFx::Movie 객체를 생성할 필요는 없다. 따라서 개발자는 관련된 액션스크립트 오버헤드 없이 직접적으로 뷰포트의 텍스트 필드를 조작하는 것이 가능하다.

DrawText API 에는 2 가지 폰트를 사용 할수 있는데, 각각 시스템폰트와 SWF/GFX 에서 로드 되는 폰트다. 시스템폰트는 시스템 폰트 제공자를 사용해야 하고, SWF 파일의 폰트는 GFx::MovieDef (GFx::Loader::CreateMovie 를 사용함)형태로 로드 되어야 한다. 그리고 나서 GFx::DrawTextManager 의 전달인자로 GFx::MovieDef 를 사용할 수 있다.

C++에서는 GFx::DrawTextManager 와 GFx::DrawText 클래스가 텍스트 렌더링 기능을 제공하는데, 다음과 같은 용도로 사용되며, 자세한 설명은 본 문서에서 차근차근 하도록 하겠다.

- *GFx::DrawTextManager* - DrawText 클래스의 인스텐서를 생성한다. 텍스트 렌더링, 뷰포트, 폰트 설정, 텍스트 렌더링 시작/중지 등의 용도로 사용

- *GFx::DrawText* –화면상의 텍스트 필드 각각의 사각영역을 나타낸다. 텍스트 포매팅과 렌더링 기능을 가지고 있다. 플래시 HTML 태그를 파싱하거나 추가적인 포매팅 데이터에 기반해서 일반 텍스트를 사용하는 것도 가능하다. SetColor, SetFont, SetFontStyle 등의 멤버함수를 사용해서 텍스트 필드의 특성값을 바꾸는 것이 가능하다.

1.1 GFx::DrawTextManager

DrawTextManager 클래스의 인스턴스는 DrawText 클래스의 인스턴스들을 관리한다. 즉, 생성, 렌더링, 텍스트 크기 측정 등의 기능 말이다.

DrawTextManager 클래스의 인스턴스를 생성하는 방법은 몇 가지가 있다.

1. DrawTextManager();

디폴트 생성자를 사용하면 내부적인 폰트캐시 메니저와 리소스 라이브러리, 로그등의 인스턴스를 생성한다. 그리소 시스템 폰트 제공자가 사용된다(SetFontProvider 메소드)

사용예:

```
Ptr<DrawTextManager> pdm = *new DrawTextManager();
```

2. `DrawTextManager(MovieDef* pmovieDef);`

이 생성자는 MovieDef 전달인자를 필요로 한다. DrawTextManager 의 인스턴스는 전달된 MovieDef 인스턴스의 모든 상태를 상속받는다(폰트 캐시 매니저, 폰트 제공자 등). MovieDef 인스턴스에 포함된 모든 폰트가 DrawTextManager 에서 사용 가능하다(이 DrawTextManager 로 생성된 DrawText 에서도 사용 가능하게 된다).

사용예:

```
Ptr<GFx::MovieDef> pmovieDef = *mLoader.CreateMovie("drawtext_fonts.swf",
Loader::LoadAll);
```

```
Ptr<GFx::DrawTextManager> pDm2 = *new DrawTextManager(pmovieDef);
```

3. `DrawTextManager(GFx::Loader* ploader);`

이 생성자는 GFx::Loader 포인터를 전달 인자로 필요로 하며 폰트 라이브러리 등의 Loader를 설정할 모든 상태를 상속받는다.

사용예:

```
GFx::Loader mLoader;
..
Ptr<GFx::DrawTextManager> pDm1 = *new DrawTextManager(&mLoader);
```

1.1.1. DrawText 생성

DrawTextManager 클래스 내에는 DrawText 인스턴스를 생성하기 위한 몇가지 메소드가 존재한다. 이때, DrawText 인스턴스를 몇 개를 생성하던 DrawTextManager 는 하나만 있으면 된다.

- `DrawText* CreateText(const char* putf8Str, const RectF& viewRect, const TextParams* ptxtParams = NULL, unsigned depth = ~0u);`
- `DrawText* CreateText(const wchar_t* pwstr, const RectF& viewRect, const TextParams* ptxtParams = NULL, unsigned depth = ~0u);`
- `DrawText* CreateText(const String& str, const RectF& viewRect, const TextParams* ptxtParams = NULL, unsigned depth = ~0u);`

위에 있는 메소드들은 일반 텍스트(NULL 문자로 끝나는 UTF-8 문자열, NULL 문자로 끝나는 유니코드/UCS-2, Scaleform::String)를 사용해서 DrawText 인스턴스를 생성한다.

`putf8str` : NULL 문자로 끝나는 UTF-8 문자열

`pwstr` : NULL 문자로 끝나는 유니코드/UCS-2

`str`: Scaleform::String

`viewRect` : 픽셀단위의 텍스트 뷰 영역 좌표, 좌측상단 위치(BeginDisplay 참고)

`ptxtParams` : 추가 전달인자. 새롭게 생성된 텍스트에 대한 전달값. TextParams 은 다음과 같다.

```
struct TextParams
{
    Color               TextColor;
    DrawText::Alignment  HAlignment;
    DrawText::VAlignment VAlignment;
    DrawText::FontStyle  FontStyle;
```

```
    float               FontSize;
    String              FontName;
    bool                Underline;
    bool                Multiline;
    bool                WordWrap;
};
```

TextColor : 텍스트의 컬러, 알파채널 포함

HAlignment: 가로정렬. 가능한 값(DrawText::Align_Left (default), DrawText::Align_Right, DrawText::Align_Center, DrawText::Align_Justify.)

VAlignment : 세로정렬,가능한값
(DrawText::VAlign_Top (default), DrawText::VAlign_Bottom, DrawText::VAlign_Center)

FontStyle : 볼드, 이탤릭 등의 폰트 스타일 지정, 가능한 값(DrawText::Normal, DrawText::Bold, DrawText::Italic, DrawText::BoldItalic (DrawText::ItalicBold 와 동일)).

FontSize: 픽셀단위 폰트크기, 실수값 가능
FontName:폰트명, "Arial", "Times New Roman"등의 형태로 사용. "Bold", "Italic"등의 스타일 지정은 불가하므로 FontStyle 을 사용할 것.

Underline: 밑줄을 사용할 것인지 지정

Multiline: 텍스트의 여러줄,한줄 모드 지정

WordWrap:워드랩핑(wordwrapping) 지정, 여러줄 텍스트 박스일 경우에만 유효

ptxtParams 추가 전달인자가 지정되지 않으면 DrawTextManager 는 기본 텍스트 전달인자를 사용한다. 다음 메소드를 사용하면 이들 기본 텍스트 전달인자 값을 읽거나 쓸 수 있다.

```
void SetDefaultTextParams(const TextParams& params);
const TextParams& GetDefaultTextParams() const;
```

다음 DrawTextManager 메소드(즉, CreateHtmlText)는 앞서 설명한 CreateText 와 비슷하지만, HTML 에서 DrawText 인스턴스를 생성한다는 것이 다르다.

```
    // Creates and initialize a DrawText object using specified HTML.

DrawText* CreateHtmlText(const char* putf8Str, const RectF& viewRect,
                    const TextParams* ptxtParams = NULL, unsigned depth = ~0u);
    DrawText* CreateHtmlText(const wchar_t* pwstr, const RectF& viewRect,
                    const TextParams* ptxtParams = NULL, unsigned depth = ~0u);
```

```
DrawText* CreateHtmlText(const String& str, const RectF& viewRect,
                         const TextParams* ptxtParams = NULL, unsigned depth = ~0u);
```

사용예:

```
// Creation of DrawText object using plain text with parameters.
String str("String No 2");
DrawTextManager::TextParams params;
params.FontName   = "Symbol";
params.FontSize   = 30;
params.FontStyle  = DrawText::Italic;
params.Multiline  = false;
params.HAlignment = DrawText::Align_Right;
params.VAlignment = DrawText::VAlign_Bottom;

Ptr<DrawText> ptxt;
ptxt = *pdm->CreateText(str, GRectF(20, 300, 400, 700), &params);

// Creation of DrawText object from HTML.
Ptr<DrawText> ptxt;
ptxt = *pdm->CreateHtmlText("<p><FONT size='20'>AB
                            <b>singleline</b><i> CD</i>O",
                            RectF(20, 300, 400, 700));
```

1.1.2. 텍스트 렌더링

텍스트는 GFx::Movie 와 비슷한 방식으로 렌더링됩니다. 텍스트 렌더링을 위해서는 DisplayHandle(GFx::DrawTextManager)을 가져오고, 뷰포트(DrawTextManager::SetViewport)를 설정할 필요가 있습니다. 뷰포트에 대한 자세한 내용은 GFx 설명서를 참조하십시오.

1.1.3. 텍스트 크기 측정

DrawTextManager 는 출력될 텍스트 사각영역의 크기를 측정하는 기능을 제공한다.

```
SizeF GetTextExtent(const char* putf8Str, float width = 0,
                            const TextParams* ptxtParams = 0);
SizeF GetTextExtent(const wchar_t* pwstr, float width = 0,
                              const TextParams* ptxtParams = 0);
SizeF GetTextExtent(const String& str, float width = 0,
                            const TextParams* ptxtParams = 0);

SizeF GetHtmlTextExtent(const char* putf8Str, float width = 0,
                                  const TextParams* ptxtParams = 0);
SizeF GetHtmlTextExtent(const wchar_t* pwstr, float width = 0,
                                 const TextParams* ptxtParams = 0);
SizeF GetHtmlTextExtent(const String& str, float width = 0,
                                 const TextParams* ptxtParams = 0);
```

GetTextExtent 는 일반 텍스트를 사용할 때 텍스트 사각영역의 크기를 계산한다. 추가적으로 TextParams 과 필요한 폭을 지정할 수 있다.
GetHtmlTextExtent 는 HTML 텍스트를 사용할 때 텍스트 사각영역의 크기를 계산한다. 추가적으로 TextParams 과 필요한 폭을 지정할 수 있다.

추가옵션인 'width'는 여러 줄이나 워드랩핑 기능을 사용할 때 필요한 텍스트 사각영역의 가로 폭을 지정하는 것이다. 이 값이 지정되면 텍스트 사각영역의 높이 값만 계산된다.

추가 옵션인 ptxtParams 는 테스트 크기를 측정할 때 사용될 텍스트 전달인자를 지정한다. 값이 지정되지 않으면 기본 전달인자 값이 사용된다. (SetDefaultTextParams / GetDefaultTextParams 참조) HTML 의 경우에는 ptxtParams 의 값이 기본 텍스트 전달인자처럼 작동하게 되는데, 즉, HTML 에서 파싱된 스타일이 가상적으로 ptxtParams 값에 덮어 써지게 되는 셈이다.

사용예:

```
 // Plain text extents
String str("String No 2");
DrawTextManager::TextParams params;
params.FontName   = "Symbol";
params.FontSize   = 30;
params.FontStyle  = DrawText::Italic;
params.Multiline  = false;
params.HAlignment = DrawText::Align_Right;
params.VAlignment = DrawText::VAlign_Bottom;

SizeF sz = pdm->GetTextExtent(str, 0, params);

params.WordWrap  = true;
params.Multiline = true;
sz = pdm->GetTextExtent(str, 120, params);

// HTML text extents
 const wchar_t* html =
    L"<p><FONT size='20'>AB <b>singleline</b><i> CD</i>O";
sz = pdm->GetHtmlTextExtent(html);

sz = pdm->GetHtmlTextExtent(html, 150);
```

1.2 GFx::DrawText

DrawText 클래스는 텍스트를 세팅하고, HTML 을 파싱하고, 포매팅하고, 렌더링하는 기능을 제공한다.

1.2.1 일반 텍스트 설정(set), 얻기(get) 메소드

SetText 메소드는 UTF-8, UCS-2, Scaleform::String 텍스트를 텍스트 오브젝트에 설정한다. 추가 전달인자인 'lengthInBytes'는 UTF-8 문자열의 바이트 수를 나타내며, 'lengthInChars'는 wide 문자열의 문자수를 나타낸다. 이들 전달인자가 지정되지 않았다면, UTF-8 나 UCS-2 는 NULL 문자로 종료되는 문자열이어야 한다.

```
void SetText(const char* putf8Str, UPInt lengthInBytes = UPInt(-1));
void SetText(const wchar_t* pstr,  UPInt lengthInChars = UPInt(-1));
void SetText(const String& str);
```

GetText 메소드는 현재 설정된 텍스트를 UTF-8 형태로 가져온다. 이 메소드는 일반 텍스트를 반환하므로, HTML 이 사용되었어도, HTML 테그가 제거된 문자열이 반환된다.

```
String GetText() const;
```

1.2.2 HTML 텍스트 설정(set), 얻기(get) 메소드

SetHtmlText 메소드는 HTML 로 인코딩된 UTF-8, UCS-2, Scaleform::String 을 파싱하고 초기화 한다. 추가 전달인자인 'lengthInBytes'는 UTF-8 문자열의 바이트 수를 나타내며, 'lengthInChars'는 wide 문자열의 문자수를 나타낸다. 이들 전달인자가 지정되지 않았다면, UTF-8 나 UCS-2 는 NULL 문자로 종료되는 문자열이어야 한다.

```
void SetHtmlText(const char* putf8Str, UPInt lengthInBytes = UPInt(-1));
void SetHtmlText(const wchar_t* pstr,  UPInt lengthInChars = UPInt(-1));
void SetHtmlText(const String& str);
```

GetHtmlText 메소드는 현재 설정된 텍스트를 HTML 형태로 가져온다. 만약 일반텍스트에 SetColor, SetFont 등의 함수를 사용해서 포매팅이 되어 있다면, 적절한 HTML 포맷으로 변형된다.

```
String GetHtmlText() const;
```

1.2.3 텍스트 포맷 설정 메소드

텍스트 포맷을 설정하거나 얻어오는 메소드 들이 있다.

```
void SetColor(Color c, UPInt startPos = 0, UPInt endPos = UPInt(-1));
```
전체 텍스트 혹은 지정된 구간[startPos..endPos]의 텍스트를 컬러 (R, G, B, A)색으로 설정한다. 'startPos' 와 'endPos'는 선택적으로 지정가능하다.

```
void SetFont (const char* pfontName, UPInt startPos = 0, UPInt endPos = UPInt(-1))
```
전체 텍스트 혹은 지정된 구간[startPos..endPos]의 텍스트 폰트를 설정한다. 'startPos' 와 'endPos'는 선택적으로 지정가능하다.

```
void SetFontSize(float fontSize, UPInt startPos = 0, UPInt endPos = UPInt(-1))
```
전체 텍스트 혹은 지정된 구간[startPos..endPos]의 텍스트 폰트크기를 설정한다. 'startPos' 와 'endPos'는 선택적으로 지정가능하다.

```
void SetFontStyle(FontStyle, UPInt startPos = 0, UPInt endPos = UPInt(-1));

   enum FontStyle
   {
      Normal,
      Bold,
      Italic,
      BoldItalic,
      ItalicBold = BoldItalic
   };
```
전체 텍스트 혹은 지정된 구간[startPos..endPos]의 폰트 스타일을 설정한다. 'startPos' 와 'endPos'는 선택적으로 지정가능하다.

```
void SetUnderline(bool underline, UPInt startPos = 0, UPInt endPos = UPInt(-1))
```
전체 텍스트 혹은 지정된 구간[startPos..endPos]의 밑줄을 설정한다. 'startPos' 와 'endPos'는 선택적으로 지정가능하다.

```
void SetMultiline(bool multiline);
```
텍스트를 여러줄(true)이나 한줄(false)로 설정한다.

```
bool IsMultiline() const;
```
텍스트가 여러줄이면 true 를 아니면 false 를 반환한다.

```
void SetWordWrap(bool wordWrap);
```
워드랩핑을 켜고 끈다.

```
bool Wrap() const;
```
워드랩핑 상태를 반환한다.

1.2.4 텍스트 정렬 메소드

Alignment 그리고 VAlignment 타입 정의

```
enum Alignment
{
   Align_Left,
   Align_Default = Align_Left,
   Align_Right,
   Align_Center,
   Align_Justify
};
enum VAlignment
{
   VAlign_Top,
   VAlign_Default = VAlign_Top,
   VAlign_Center,
   VAlign_Bottom
};
```

텍스트 정렬 메소드 들은 다음과 같다.

`void SetAlignment(Alignment);`
텍스트 수평 정렬 설정(right, left, center).

`Alignment  GetAlignment() const;`
텍스트 수평 정렬 반환(right, left, center).

`void SetVAlignment(VAlignment);`
텍스트 수직 정렬 설정(top, bottom, center).

`VAlignment  GetVAlignment() const;`
텍스트 수직 정렬 반환(top, bottom, center).

1.2.5 텍스트 위치와 방위 변경

DrawText 클래스는 위치와 방위를 위한 다양한 메소드를 제공한다.

`void SetRect(const RectF& viewRect);`
뷰 사각영역 설정, 좌표계는 픽셀

`RectF  GetRect() const;`
사용중인 뷰 사각영역 얻기, 좌표계는 픽셀

`void SetMatrix(const Matrix& matrix);`
텍스트의 변환행렬 설정

```
const Matrix GetMatrix() const;
```
텍스트의 변환행렬 반환

```
void SetCxform(const Cxform& cx);
```
텍스트의 컬러변환행렬 설정

```
const Cxform& GetCxform() const;
```
텍스트의 변환행렬 얻기

2 DrawText 예제코드 요약

이번 장에서는 Scaleform 4.1 에 포함된 DrawText API 예제 코드를 설명할 것이다. DrawText API 는 Scaleform SDK 에 포함되어 배포되는 것으로 간단한 텍스트 오브젝트를 메모리나 퍼포먼스 오버헤드 없이 화면에 출력해 준다. 예제코드는 DrawText API 에서 텍스트를 그리는데 사용된 클래스에 대한 설명이 될 수 있으며, 당신의 텍스트 엔진에 손쉽게 포함될 수 있다.

기본 단계에는 창 설정, 무비 데이터 로드, 캡쳐를 위한 디스플레이 핸들 추가, 텍스트 캡처 및 렌더링이 포함됩니다.

FxPlayerApp 클래스는 창을 설정하고 동영상을 불러오고 텍스트를 렌더링하기 위한 모든 속성과 함수를 정의하는 플레이어 응용 프로그램 클래스입니다. FxPlayerApp 은 창, 렌더 스레드 및 그래픽을 생성하고 초기화하는 기본 클래스인 FxPlayerAppBase 클래스를 상속합니다.

FxPlayerApp 의 OnInit 메서드는 DrawText 인스턴스를 초기화하고 텍스트를 만들고 다양한 텍스트 변형을 적용합니다. OnUpdateFrame 메서드는 텍스트에 변형 매트릭스를 적용하여 애니메이션을 만듭니다.

```
class   FxPlayerApp : public FxPlayerAppBase
{
    public:
    FxPlayerApp();

    virtual bool          OnInit(Platform::ViewConfig& config);
    virtual void          OnUpdateFrame(bool needRepaint);

    Ptr<GFx::DrawTextManager>      pDm1, pDm2;
    Ptr<GFx::DrawText>         ptxt11, ptxt12,  ptxt21, ptxt22, ptxtImg,
                            pblurTxt, pglowTxt, pdropShTxt, pblurglowTxt;
    float   Angle;
    UInt32  Color;
};
```

DrawText 예제는 DrawTextManager 클래스 인터페이스의 2 개의 다른 인스턴스를 사용해서 어떻게 텍스트를 렌더링 할 지 보여준다.

방법 1

이 경우에는 DrawTextManager 가 로더로부터 모든 상태를 상속받은 GFx::Loader 를 사용해서 초기화 되었을때다.

```
pDm1 = *new DrawTextManager(&mLoader);
```

여기에서는 텍스트를 시스템 폰트로 표시하기 위해 GFx::FontProvider 를 사용합니다. FontProvider 는 FxPlayerAppBase::OnInit 에서 생성됩니다.

```
pDm1->SetFontProvider(mLoader.GetFontProvider());
```

API 에서 언급된 것처럼 DrawText 는 일반 텍스트나 HTML 로 생성가능하다. 여기서는 Scaleform::String 과 일반 텍스트를 사용해서 초기화 하였다.

```
// Create text using plain text and default text parameters.
DrawTextManager::TextParams defParams =
    pDm1->GetDefaultTextParams();
defParams.TextColor = Color(0xF0, 0, 0, 0xFF); // red, alpha = 255
defParams.FontName  = "Arial";
defParams.FontSize  = 16;
pDm1->SetDefaultTextParams(defParams);
```

텍스트는 CreateText 를 사용해서 생성되며, 이때 앞서 호출한 SetDefaultTextParams 의 기본 텍스트 전달인자를 사용한다.

```
ptxt11 = *pDm1->CreateText ("Plain text, red, Arial,
                            16pts",RectF(20, 20, 500, 400));
```

DrawTextManager 클래스의 GetTextExtent 메서드는 텍스트 사각형의 크기를 측정합니다. DrawText 클래스는 폰트 형식을 지정하고 텍스트를 정렬하고 텍스트 객체의 위치를 바꾸는 등 텍스트 관련 작업을 할 때 유용합니다.

```
// Create text with using String and TextParams
String str(
  "Scaleform GFx is a light-weight high-performance rich media vector
   graphics and user interface (UI) engine.");
GFx::DrawTextManager::TextParams params;
params.FontName   = "Arial";
params.FontSize   = 14;
params.FontStyle  = DrawText::Italic;
params.Multiline  = true;
params.WordWrap   = true;
params.HAlignment = DrawText::Align_Justify;
sz = pDm1->GetTextExtent(str, 200, &params);
```

```
ptxt12 = *pDm1->CreateText(str, RectF(200, 300, sz), &params);
ptxt12->SetColor(Render::Color(0, 0, 255, 130), 0, 1);
```

시스템 폰트로 텍스트를 만든 다음 DrawText 의 뷰포트를 설정하고 현재 DrawText 상태를 캡처한 뒤에 DrawText DisplayHandle 을 RenderThread 에 추가합니다.

```
Render::Size<unsigned> viewSize = GetViewSize();
Render::Viewport dmViewport(viewSize.Width, viewSize.Height,
          int(viewSize.Width * GetSafeArea().Width),
          int(viewSize.Height * GetSafeArea().Height),
          int(viewSize.Width - 2 * viewSize.Width * GetSafeArea().Width),
          int(viewSize.Height - 2 * viewSize.Height * GetSafeArea().Height));
   pDm1->SetViewport(dmViewport);
pDm1->Capture();
pRenderThread->AddDisplayHandle(pDm1->GetDisplayHandle(),
                           FxRenderThread::DHCAT_Overlay, false);
```

방법 2

이번 예제는 DrawTextManger 인스턴스 생성시에 폰트 정보를 포함하고 있는 GFx::MovieDef 의 포인터를 사용한다. MovieDef 에 로딩된 SWF 파일로부터 폰트를 얻어낸다. Scaleform 와 무비 로딩에 관련된 사항은 Scaleform Integration Tutorial 을 참고하도록 하자.

```
Ptr<MovieDef> pmovieDef = *mLoader.CreateMovie("drawtext_fonts.swf",
                                               Loader::LoadAll);
      pDm2 = *new DrawTextManager(pmovieDef);
```

앞서 예제와 다르게, 일반 텍스트가 아닌 HTML 텍스트를 이용해서 출력한다. 텍스트 사각영역은 **DrawTextManager::GetHtmlExtent** 를 사용해서 계산한다.

```
// Create HTML text, using fonts from GFx::MovieDef
const wchar_t* html = L"<P>o123 <FONT FACE=\"Times New Roman\" SIZE
            =\"140\">"L"A<b><i><FONT
      COLOR='#3484AA'>б</FONT></i>рак</b>адабрА!</FONT></P>"
      L"<P><FONT FACE='Arial Unicode MS'>Hànyǔ; 华语/華語</FONT></P>"
      L"<P><FONT FACE='Batang'>한국어/조선말</FONT></P>"
      L"<P><FONT FACE='Symbol'>Privet!</FONT></P>";

GFx::DrawTextManager::TextParams defParams2 = pDm2->GetDefaultTextParams();
defParams2.TextColor = Color(0xF0,0,0,0xFF); //red,alpha = 255
defParams2.Multiline = true;
defParams2.WordWrap  = false;
SizeF htmlSz = pDm2->GetHtmlTextExtent(HtmlText, 0, &defParams2);
```

```
ptxt22 = *pDm2->CreateHtmlText(HtmlText, RectF(00, 100, htmlSz),
&defParams2);
```

또한 이번에는 텍스트에 애니메이션(회전과 컬러변환 매트릭스)을 넣어서 생성하였다.

```
SizeF sz;
sz = pDm2->GetTextExtent("Animated");
ptxt21 = *pDm2->CreateText("Animated", RectF(600, 400, sz));
ptxt21->SetColor(Render::Color(0, 0, 255, 255));
Angle = 0;
```

필터:

```
SizeF sz;
Ptr<Scaleform::DrawText>  pglowTxt
pglowTxt = *pDm2->CreateText("Glow", RectF(800, 350, SizeF(200, 220)));
pglowTxt->SetColor(Render::Color(120, 30, 192, 255));
pglowTxt->SetFontSize(72);
Scaleform::DrawText::Filter glowF(Scaleform::DrawText::Filter_Glow);
glowF.Glow.BlurX = glowF.Glow.BlurY = 2;
glowF.Glow.Strength = 1000;
glowF.Glow.Color = Render::Color(0, 0, 0, 255).ToColor32();
pglowTxt->SetFilters(&glowF);
```

다음과 같이 DrawText 의 뷰포트를 설정하고 현재 DrawText 상태를 캡처하며 DrawText DisplayHandle 을 RenderThread 에 추가합니다.

```
pDm2->SetViewport(dmViewport);
pDm2->Capture();
pRenderThread->AddDisplayHandle(pDm2->GetDisplayHandle(),
                                FxRenderThread::DHCAT_Overlay, false);
```

DrawText 에서 제공하는 컬러와 위치 변환을 위해서 다른 함수를 사용해 보자.
DrawText 의 **SetMatrix** 와 **SetCxform** 을 사용해서 텍스트의 회전과 컬러 변화를 주고 있다**.** 이 작업은 FxPlayerApp::OnUpdateFrame 에서 수행합니다.

```
void FxPlayerApp::OnUpdateFrame( bool needRepaint )
{
    DrawText::Matrix txt21matrix;
    Angle += 1;
    RectF r = ptxt21->GetRect();
    txt21matrix.AppendTranslation(-r.x1, -r.y1);
    txt21matrix.AppendRotation(Angle*3.14159f / 180.f);
    txt21matrix.AppendScaling(2);
    txt21matrix.AppendTranslation(r.x1, r.y1);
    ptxt21->SetMatrix(txt21matrix);
```

```
        pDm2->Capture();

        FxPlayerAppBase::OnUpdateFrame(needRepaint);
    }
```

주의: Bin/Samples 폴더의 drawtext_fonts.swf 를 실행파일과 동일한 폴더나 프로젝트 폴더에 복사해야 한다(Visual Studio 2008 이라면 Projects/Win32/Msvc90/Samples/DrawText 폴더). 이렇게 하지 않으면 대부분의 그냥 단순 사각형만 보게 될 것이다.

실행화면: