Autodesk®
Scaleform®

Scaleform 3D

이 문서는 Scaleform 3.2 및 상위 버전에서 제공하는 3D 기능을 사용하는 방법을 기술한 문서입니다.

작성자: Mustafa Thamer

버전: 2.03

최종 수정일: 2012 년 5 월 9 일

Autodesk®
GAMEWARE

Copyright Notice

Autodesk® Scaleform® 4.4

Disclaimer

Autodesk Scaleform 연락 방법

문서	Scaleform 3D
주소	Scaleform Corporation
	6305 Ivy Lane, Suite 310
	Greenbelt, MD 20770, USA
홈페이지	www.scaleform.com
전자 우편	info@scaleform.com
직통 전화	(301) 446-3200
팩스	(301) 446-3199

차례

1 개요

Scaleform® 3Di™이 제공하는 새로운 3D Flash 렌더링 및 애니메이션 기능으로 개발자 여러분들은 새로운 차원의 응용 프로그램 UI 를 구현하실 수 있습니다.

Scaleform 3.2 에서 처음 선보인 이후 Scaleform 은 기본적인 3D 기능을 제공하는 ActionScript 익스텐션으로 구성된 간단하지만 강력한 도구 모음을 추가해왔습니다. Scaleform 3.2 는 ActionScript 2.0 를 기반으로 하지만, 이후 AS 3.0 에서 제공하는 기능에 준하는 3D AS 익스텐션을 계속적으로 추가 제공해왔습니다.

이러한 새 익스텐션(_z, _zscale, _xrotation, _yrotation, _matrix3d, _perspfov)을 사용하면, 메뉴, HUD 및 게임 UI 에 적합한 놀라운 3D 애니메이션 인터페이스를 구현할 수 있습니다. 이러한 3D 익스텐션은 어떤 movieclip, button 및 textfield 오브젝트에도 적용 가능합니다.

동일한 3D 기능을 Scaleform Direct Access API 의 일부로서 C++에서 사용할 수 있습니다. 새 API 를 통해 개발자들은 직접 3D 내에 GFx::Value 로 할당된 movieclip, button, textfield 를 크기 조정, 회전 및 오프셋 등으로 변경할 수 있습니다.

게다가, 자체적으로 렌더링한 3D 오브젝트 또는 스트리밍 비디오를 3D Flash 인터페이스에 추가하여 독특하고 발전된 인터페이스를 구성할 수도 있습니다.

2 Flash 10 과의 유사성

일반적으로 Scaleform 에서 제공하는 3Di 지원은 Flash 10 의 3D 와 유사하며 같은 방식으로 동작하도록 되어 있습니다. 3Di 는 Scaleform 에 포함된 AS2 익스텐션 기본 셋을 사용하여 실행됩니다.

2.1 제한 사항

Flash 10 에서의 3D 지원과 유사한 3Di 에는 다음과 같은 제한 사항이 있습니다.

1. 깊이 정렬을 지원하지 않아 오브젝트가 올바른 순서로 그려지지 않을 수 있습니다. 그리는 순서는 여전히 기본적인 레이어 순서에 따라 지정됩니다.
2. 후면 추려내기(Back-face culling)가 사용되지 않습니다. 뷰어에서 빗나간 오브젝트는 그려집니다.

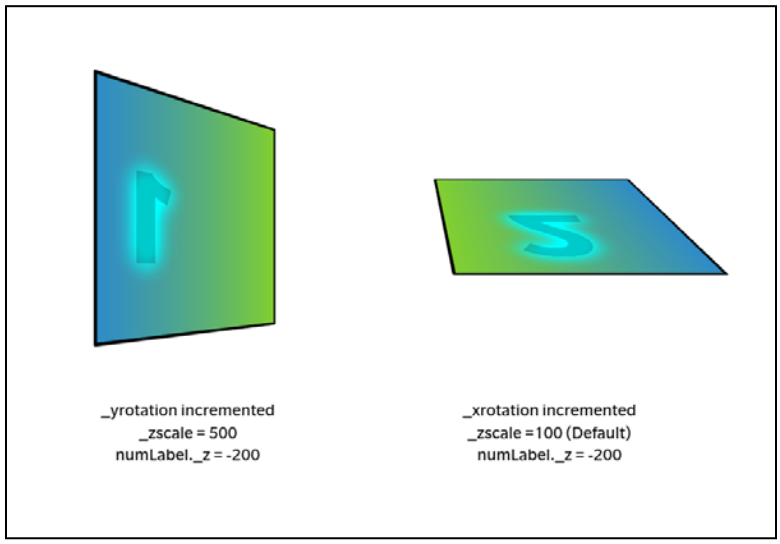

그림 1: 추려내지지 않았거나 후방에 그려진 후면 숫자의 예

2.2 실행

Flash 10 이 비트맵 내부로 3D 뷰를 렌더링한 다음 화면을 채우는 용도로 해당 비트맵을 벡터 트라이앵글과 함께 사용하는 반면, Scaleform 3Di 는 하드웨어 가속 트라이앵글을 사용하여 셰이프를 렌더링합니다.

오브젝트가 2D 와 3D 속성을 둘 다 가질 경우, 3D 속성이 2D 속성에 우선하여 적용됩니다. 이에 대한 세부 사항은 3D 연산은 물론 자유 변형 도구로 오브젝트를 회전시킬 때 원하는 결과를 얻기 위해서 중요하므로 반드시 이해하도록 합니다. Flash 10 에서도 동일합니다.

3 3D 에서 Flash 출력하기

Scaleform 3.2를 사용하여 3D로 Flash 콘텐츠를 출력하는 데 사용할 수 있는 방법에는 2가지가 있습니다.

3.1 텍스처에 렌더링

첫 번째는 전통적인 방법으로, 텍스처 내에 Flash를 렌더링하고 그 다음에 해당 장면의 3D 서페이스 위에 지도와 같은 렌더링 텍스처를 적용하는 것입니다. Scaleform 3.2 이전 버전에서는 3D에서 Flash를 출력할 수 있는 유일한 방법이었습니다.

텍스처에 렌더링하는 방법은 3Di보다 유리한 점이 있습니다. Flash 콘텐츠가 3D 오브젝트에 텍스처 매핑되었기 때문에, 3D 엔진의 성능을 최대로 활용할 수 있으며, 나머지 3D 환경과 심리스하게 혼합됩니다. 이 방법은 올바른 Z(깊이) 정렬, 블렌딩/투명도 및 클리핑(UI 클리핑에 대한 의도와 관계 없음)을 허용합니다.

반면, 렌더링 텍스처 때문에 추가 메모리를 요구하므로 성능 및 메모리에 관한 몇 가지 단점들이 있습니다. 또한, 모든 Flash 상호 작용은 게임에 의해서(수동으로 마우스 좌표를 인버스 매핑하여 Scaleform로 넘김) 제어되어야 합니다. 마지막으로, 이러한 접근은 텍스처 기반이므로 Flash 콘텐츠가

평면으로 출력됩니다. 따라서 개별적인 movie clip을 각각 오프셋하거나 회전할 수 없습니다. 이 접근법의 예는 'SWF to Texture' 데모를 참조하십시오. 해당 데모는 Scaleform SDK Browser에서 찾을 수 있습니다.

3.2 3Di

두 번째 방법은 Scaleform 3.2에서 새로 선보이는 방법으로, ActionScript 또는 Direct Access API(C++)를 통해서 3Di를 사용하는 것입니다. 3Di는 렌더링 텍스처를 사용하는 접근법에 비해 많은 장점이 있습니다. 먼저, 추가 메모리가 필요하지 않습니다. 또한, 각각의 movie clip을 개별적으로 변형할 수 있습니다. 모든 movie clip을 개별적으로 오프셋하거나 회전할 수 있으므로 전체 movie가 평면으로 보이지 않습니다. 마지막으로, 모든 입력 및 상호 작용은 Scaleform에 의해 자동으로 제어되어, 응용 프로그램이 보통 때처럼 간단하게 Scaleform로 입력 이벤트를 전달합니다.

하지만, 3Di에는 몇 가지 제한 사항이 있습니다. 특히, 기본 Scaleform 렌더러로 제어되며 UI로 다뤄지기 때문에 깊이 정렬이나 후면 추려내기(back-face culling)가 수행되지 않습니다. 따라서, UI 오브젝트가 다른 오브젝트 후방에서 회전하는 것처럼 올바른 순서로 그려지지 않습니다. 또한, UI 오브젝트는 같은 카메라, 빛, 전처리 및 사용 중인 다른 그래픽 효과에 맞춰야 하므로, 3D 게임 오브젝트를 가지는 심리스한 융합 UI 구성 요소에는 문제가 있을 수 있습니다. 또한, 많은 UI 구성 요소가 투명하며 블렌딩 때문에 정확한 그리기 순서를 요구하는데, 이런 점들을 고려에 넣어야 합니다.

4 3Di API

4.1 ActionScript 2 API

이제 Flash 오브젝트는 3Di 를 통해 2D 에서 3D 로 변신할 수 있습니다. 이런 변화를 통해, 사용자들은 시점을 앞뒤로 움직여가면서 3D 화된 오브젝트를 볼 수 있습니다. 오브젝트가 카메라에서 멀어지면 원근법 투사에 따라 오브젝트의 크기가 작아집니다.

기본적으로 오브젝트는 2 차원이며 3D 속성은 사용되지 않거나 계산되지 않습니다. 하지만, 3D 속성이 설정되면(x 또는 y 회전축, z 이동축, 스케일 또는 3d 매트릭스), 오브젝트가 3D 로 바뀌게 됩니다. 이때, 3D 변환 매트릭스가 자동으로 생성되어 해당 오브젝트에 할당됩니다. 해당 오브젝트의 3d 매트릭스를 null 로 설정하면 2D 로 되돌아옵니다(ActionScript 에서는 `foo._matrix3d = null`). 이동축과 z 값만 0 으로 설정하면 3D 변환이 제거되지 않습니다.

앞서 언급한 것처럼, Z 축만이 회전의 중심이 되는 2D 와는 달리, 3D 회전은 3 개의 축인 x, y, z 중 어느 것이든 중심으로 회전할 수 있습니다. 비슷하게, 3D 스케일은 _zscale ActionScript 익스텐션을 통해서 크기 변경에 필요한 추가 축을 추가합니다.

Scaleform 에서 제공하는 기본 3D 좌표 시스템과 카메라는 Flash 10 에서 제공되는 것과 동일합니다. 오른쪽으로만 회전하는 시스템으로, +X 는 오른쪽, +Y 는 아래쪽이며 +Z 는 화면으로 이동(뷰어에서 멀어짐)합니다. 기본 카메라는 55 단계의 FOV 각도로 +Z 축을 내려다 봅니다.

3D 속성을 적용할 수 있는 Flash 오브젝트 형식은 다음과 같습니다.

- Movie clips
- Text fields
- Buttons

3Di 에는 다음 ActionScript 익스텐션이 추가되었습니다.

- **_z** - 오브젝트의 Z 좌표(깊이)를 설정합니다. 초기값은 0 입니다.
- **_zscale** - 오브젝트의 스케일을 Z 방향 쪽으로 해서 백분율로 표시합니다. 초기값은 0 입니다.
- **_xrotation** - X(수평)축을 중심으로 오브젝트의 회전을 설정합니다. 초기값은 0 입니다.
- **_yrotation** - Y(수직)축을 중심으로 오브젝트의 회전을 설정합니다. 초기값은 0 입니다.
- **_matrix3d** - 16 개의 float 로 구성된 배열(4 x 4)를 사용하여 오브젝트의 3D 변환을 설정합니다. 값이 NULL 로 설정되면 모든 3D 변환이 제거되고 오브젝트가 2D 로 바뀝니다.
- **_perspfov** –객체의 원근 시야(Field Of View)각을 설정합니다. 각도 값은 0 보다 크고 180 보다 작거나 원근을 비활성화하고 직각 시야를 사용할 경우 -1 이 되어야 합니다. 이 값을 설정하지 않으면 객체는 루트 FOV 각도(초기값 55)를 상속받습니다.

이러한 새 익스텐션을 사용하려면 전역 변수인 gfxExtensions 가 ActionScript 에서 true 로 설정되어야 합니다.

movie clip 인스턴스의 회전을 수정하려면 등록 지점의 위치를 기록해 두어야 합니다. movie clip 심볼을 생성할 때 등록 포인트는 기본적으로 상단 왼쪽 모서리로 설정됩니다. _xrotation 또는 _yrotation 를 사용해서 movie clip 에 3D 회전을 적용하면 오브젝트는 등록 지점을 중심으로 회전하게 됩니다. 원한다면 등록 지점을 심볼의 중심으로 설정할 수 있습니다. 3Di 를 사용해서 오브젝트에 회전을 적용하려면 주의해서 등록 지점을 설정해야 합니다.

예 1

Y 축을 중심으로 movie clip 을 45 도 회전시킵니다.

```
_global.gfxExtensions = true;
mc1._yrotation = 45;
```

예 2

Z 스케일을 따라서 X 축을 중심으로 계속 회전하도록 합니다.

```
_global.gfxExtensions = true;
sq1._zscale = 500;

function rotateAboutXAxis(mc:MovieClip):Void
{
    mc._xrotation = mc._xrotation + 1;
    if (mc._xrotation > 360)
       {
        mc._xrotation = 0;
    }
}

onEnterFrame = function()
{
    rotateAboutXAxis(sq1);
}
```

예 3

루트에 적용된 3D 이동 매트릭스를 사용하여 3D 변환을 합니다.

```
function PixelsToTwips(iPixels):Number
{
      return iPixels*20;
}


function TranslationMatrix(tX, tY, tZ):Array
{
    var matrixC:Array = [ 1, 0, 0, 0,
                0, 1, 0, 0,
                0, 0, 1, 0,
                tX,tY,tZ,1];
    return matrixC;
}

this._matrix3d = TranslationMatrix(PixelsToTwips(100),PixelsToTwips(100),0);
```

4.2 C++로 3Di 사용하기

4.2.1 Matrix4x4

4x4 의 열주도 행렬을 그리기 위해 새 클래스인 Matrix4x4 가 추가되었습니다. 이 클래스에는 다양한 유형의 3D 매트릭스(일반 월드 변환, 뷰 및 투사 등)를 생성하고 처리하는 데 적합한 모든 함수가 포함됩니다. 자세한 내용은 Render_Matrix4x4.h 를 참조하십시오.

4.2.2 Direct Access

GFx::Value 클래스를 경유하는 Direct Access 인터페이스가 3Di 지원을 위해 확장되었습니다. 이제 3D 속성은 새로운 API 와 함께 설정 혹은 질의될 수 있습니다. 이러한 함수는 3Di ActionScript 익스텐션과 유사합니다.

자세한 사항은 GFx_Player.h 내의 GFx::Value::DisplayInfo 클래스를 참조하십시오.

```
void    SetZ(Double z)
void    SetXRotation(Double degrees)
void    SetYRotation(Double degrees)
void    SetZScale(Double zscale)
void    SetFOV(Double fov)
void    SetProjectionMatrix3D(const Matrix4F *pmat)
void    SetViewMatrix3D(const Matrix3F *pmat)
bool    SetMatrix3D(const Render::Matrix3F& mat)

Double  GetZ() const
Double  GetXRotation() const
Double  GetYRotation() const
Double  GetZScale() const
Double  GetFOV() const
const Matrix4F* GetProjectionMatrix3D() const
const Matrix3F* GetViewMatrix3D() const
bool    GetMatrix3D(Render::Matrix3F* pmat) const
```

4.2.3 Render::TreeNode

영상을 3D 로 렌더링하는 데 필요한 뷰와 원근 매트릭스를 설정하기 위해 Render::TreeNode 클래스 인터페이스에 새로운 호출이 추가되었습니다. 개별 디스플레이 오브젝트에 원근을 설정할 수 있지만, 영상의 루트에 설정하면 한 번에 모든 오브젝트에 적용할 수 있으므로 편리합니다.

GFx_Player.h

```
void     SetProjectionMatrix3D(const Matrix4F& m)
void     SetViewMatrix3D(const Matrix3F& m);
```

4.2.4 예: Direct Access API 를 사용하여 원근 시야 변경하기

```
Ptr<GFx::Movie> pMovie = …;
GFx::Value tmpVal;
bool bOK = pMovie->GetVariable(&tmpVal, "_root.Window");
if (bOK)
{
      GFx::Value::DisplayInfo dinfo;
       bOK = tmpVal.GetDisplayInfo(&dinfo);
       if (bOK)
       {   // set perspectiveFOV to 150 degrees
           Double oldPersp = dinfo.GetFOV();
           dinfo.SetFOV(150);
           tmpVal.SetDisplayInfo(dinfo);
       }
}
```

5 원근

위의 예제 코드에서는 Direct Access API 를 사용하여 영상의 FOV(시야) 설정하는 방법을 알 수 있었습니다. 원근 및 뷰 설정은 영상의 루트 또는 개별적인 디스플레이 오브젝트에 적용할 수 있습니다. 오브젝트가 원근 FOV 값을 0 으로 반환하면 해당 오브젝트의 부모로부터 FOV 값을 상속받습니다.

원근은 오브젝트가 뷰어에 가까울수록 크게 보이고, 뷰어에서 멀어질수록 작게 보이도록 합니다. 시야 값을 변경하여 이 효과의 세기를 조절할 수 있습니다. 값이 커질수록 z 축을 따라 움직이는 디스플레이 오브젝트에 적용되는 왜곡이 강해집니다. FOV 값이 낮으면 크기가 매우 작아지며, 값이 크면 커다란 움직임을 보입니다. 값은 반드시 0 보다 크고 180 보다 작아야 합니다. 1 로 설정하면 원근 왜곡이 거의 없고, 179 로 설정하면 어안 렌즈 효과가 나타납니다.

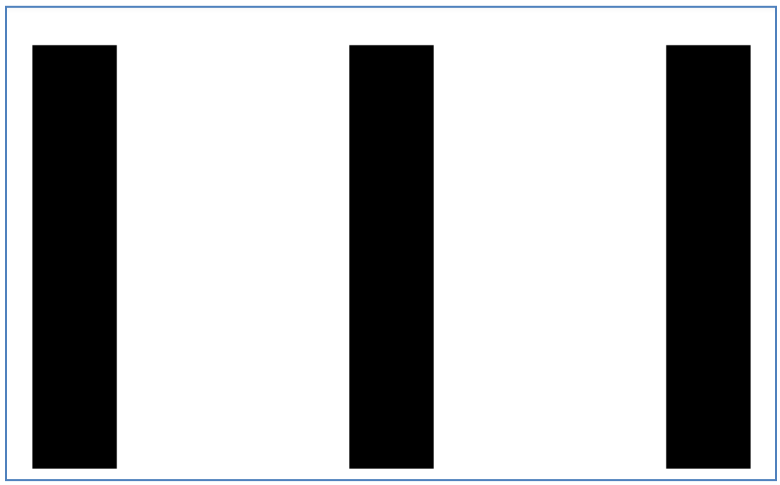

그림 2: 회전되지 않은 원래 Flash 이미지

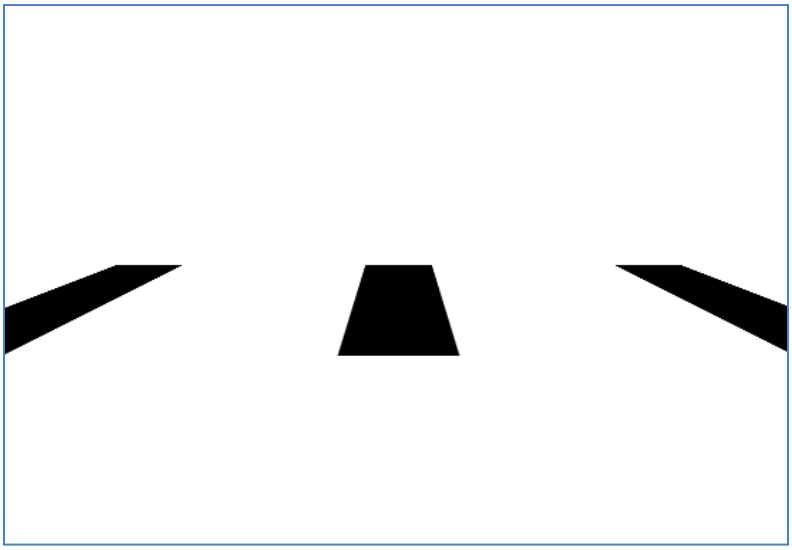

그림 3: -80 의 x 축 회전과 기본 원근 적용(55 도)

낮은 FOV 값(5 보다 낮은 값)은 종종 왜곡효과를 나타낸다. 시각 효과를 완전히 제거하기 위해서는 직각의 프로젝션을 사용해서 그 객체에 설정하는 것이 좋을 것이다. ActionScript 에서는 _perspfov 속성을 -1 로 설정하여 이를 수행할 수 있습니다. C++에서는 Direct Access API 를 이용해 다음 두 가지 방법으로 이를 수행할 수 있습니다.

// 1. 원근 FOV 를 -1 로 설정

```
GFx::Value tmpVal;
bool bOK = pMovie->GetVariable(&tmpVal, "root"); //"_root" for AS2
if (bOK)
{
   GFx::Value::DisplayInfo dinfo;
   bOK = tmpVal.GetDisplayInfo(&dinfo);
   if (bOK)
   {
      dinfo.SetPerspFOV(-1);
      tmpVal.SetDisplayInfo(dinfo);
   }
 }
```

// 2. 원근 매트릭스를 직각 매트릭스로 설정

```
void MakeOrthogProj(const RectF &visFrameRectInTwips, Matrix4F *matPersp)
{
   const float nearZ = 1;
   const float farZ = 100000;
```

```
   float DisplayHeight = fabsf(visFrameRectInTwips.Height());
   float DisplayWidth = fabsf(visFrameRectInTwips.Width());
   matPersp->OrthoRH(DisplayWidth, DisplayHeight, nearZ, farZ);
}
GFx::Value tmpVal;
bool bOK = pMovie->GetVariable(&tmpVal, "root.mc");
if (bOK)
{
   GFx::Value::DisplayInfo dinfo;
   bOK = tmpVal.GetDisplayInfo(&dinfo);
   if (bOK)
   {
      Matrix4F perspMat;
      MakeOrthogProj(PixelsToTwips (pMovie->GetVisibleFrameRect()), &perspMat);
      dinfo.SetProjectionMatrix3D(&perspMat);
      tmpVal.SetDisplayInfo(dinfo);
  }
}
```

영상 객체에서 직접 투사 매트릭스를 설정할 수도 있습니다.

```
Ptr<Movie> pMovie = …
// compute ortho matrix
Render::Matrix4F ortho;
const RectF &visFrameRectInTwips = PixelsToTwips(pMovie->GetVisibleFrameRect());
const float nearZ = 1;
const float farZ = 100000;
ortho.OrthoRH(fabs(visFrameRectInTwips.Width()),  fabs(visFrameRectInTwips.Height()),
nearZ, farZ);
pMovie->SetProjectionMatrix3D(ortho);
```

5.1 AS3 의 원근 설정

ActionScript 3 에서 PerspectiveProjection 오브젝트를 사용하여 오브젝트의 원근을 조정할 수 있습니다.

Example:

```
import flash.display.Sprite;

var par:Sprite = new Sprite();
par.graphics.beginFill(0xFFCC00);
par.graphics.drawRect(0, 0, 100, 100);
par.x = 100;
```

```
par.y = 100;
par.rotationX = 20;

addChild(par);

var chRed:Sprite = new Sprite();
chRed.graphics.beginFill(0xFF0000);
chRed.graphics.drawRect(0, 0, 100, 100);
chRed.x = 50;
chRed.y = 50;
chRed.z = 0;

par.addChild(chRed);

var pp3:PerspectiveProjection=new PerspectiveProjection();
pp3.fieldOfView=120;
par.transform.perspectiveProjection = pp3;
```

6 스테레오스코픽 3D

Scaleform 3Di 는 스테레오 3D 이미징을 채용한 게임과 응용 프로그램의 UI 를 생성하는 데 사용할 수 있습니다. 일반적으로 스테레오 기술에는 양쪽 눈에 맞춰서 가볍게 오프셋 처리한 두 개의 이미지를 생성해 깊이에 대한 환각 효과를 향상시키는 방법이 포함됩니다. 이러한 두 개의 이미지는 같은 장면에 대한 두 개의 원근을 나타내며, 뇌가 깊이를 느끼도록 해줍니다. 스테레오 디스플레이 장치와 연동되는 특수 안경을 쓰면 각각의 눈에 맞춘 정확한 이미지가 투사됩니다.

Scaleform 3Di 는 NVIDIA 의 PC 용 3D Vision Kit 와 같은 스테레오스코픽 3D 시스템을 채용하여 놀라운 성능을 발휘합니다. NVIDIA 의 솔루션은 드라이버 단계에서 자동으로 작동하므로 코드를 변경할 필요 없이 스테레오스코피를 사용할 수 있습니다. 이것은 게임 콘솔과 같은 다른 시스템에서는 통하지 않습니다. 그러한 시스템에서의 응용 프로그램은 반드시 양쪽 눈에 맞추어 Display 를 두 번 호출하고, 좌우 눈의 시차 변화를 조절하기 위해 변형 매트릭스를 설정해야 합니다. Scaleform 4.0 에서는 이것을 지원하며 PS3 스테레오 플레이어가 TinyPlayerStereo 를 호출하는 예제를 제공합니다. TinyPlayerStereo 는 다음 Scaleform 하위 경로에 있습니다.
Apps\Samples\GFxPlayerTiny\GFxPlayerTinyPS3GcmStereo.cpp

6.1 NVIDIA 3D Vision

6.1.1 3D Vision 셋업

NVIDIA 3D Vision Kit 을 사용하여 스테레오로 3Di 를 렌더링하려면 먼저 해당 키트 및 관련 드웨어를 올바르게 설치해야 합니다. NVIDIA 컨트롤 패널로 가서. 스테레오스코픽 3D 섹션에서 '스테레오스코픽 3D 활성화'를 클릭합니다. NVIDIA 컨트롤 패널에서 Stereoscopic 3D Test 를 실행시키면 셋업이 완료됩니다.

6.1.2 실행

다음으로, 귀하의 응용 프로그램 또는 D3D9 Scaleform Player 를 전체 화면 모드로 실행합니다. Automatic 3D Vision 지원은 Direct3D Scaleform Player 의 전체 화면에서만 동작하므로 반드시 전체 화면으로 실행해야 합니다. Scaleform Player 의 경우에는 명령줄상에서 -f 를 전달하는 것이 필요합니다. NVIDIA 컨트롤 패널의 스테레오 섹션에 있는 설정 여부에 따라 3D 비활성화 상태로 응용 프로그램이을 실행할 수 있습니다. 3D 효과가 즉시 나타나지 않을 경우 (CTRL + T)를 입력하여 스테레오 모드로 전환하십시오.

6.1.3 컨버전스 프로필

스테레오 효과가 활성화되면 설정을 조절해야 합니다. 화면 깊이에 있는 오브젝트는 이탈하지 않으며 스테레오 효과도 적용되지 않습니다. 이와 비슷하게, 상대적으로 가까운 곳에 있거나 먼 곳에 있는 오브젝트는 적절하게 +이탈이나 -이탈을 하여 화면 안쪽으로 들어가거나 바깥쪽으로 튀어나온 것처럼 보이게 됩니다. 응용 프로그램에 최적화된 3D Vision 깊이와 컨버전스 값을 설정하고, 설정한 값은 향후 사용을 위해 레지스트리 프로필에 등록합니다.

일반적으로 3D Vision 만 사용할 때는 IR 송신기에서 스크롤 휠을 돌리거나 (CTRL+F3)/(CTRL+F4)을 사용해서 깊이 레벨을 줄이거나 늘리는 식으로 제어할 수 있습니다. 게임 프로필에 모든 것이 저장되기 때문에 대부분의 게임에 대해서 컨버전스를 조정할 필요가 없습니다. 사용자 정의 어플리케이션에 대해서나 Scaleform Player 를 사용할 때는 먼저 올바른 컨버전스를 설정해야 합니다. 컨버전스 레벨을 변경하는 키의 초기값은 (CTRL+F5))와 (CTRL+F6)이지만, 기본적으로 활성화되지는 않습니다. 아래 순서를 따라서 이 기능을 활성화합니다.

컨버전스의 레벨을 제어하려면 먼저 NVidia 컨트롤 패널에서 "advanced in-game settings 를 활성화"해야 합니다. 그러기 위해서는, "*Stereoscopic 3D*"로 가서 "*Set up stereoscopic 3D*"을 선택한 다음 "*Set Keyboard Shortcuts*"을 엽니다. 고급 설정이 활성화되면 (CTRL+F5)와 (CTRL+F6)를 사용해서 컨버전스를 변경할 수 있으며. (CTRL+F7)을 사용해서 선택한 사용자 정의 설정을 저장합니다. 설정을 저장하면 다음에 응용 프로그램이 실행될 때 컨버전스 설정 단계를 거치지 않아도 됩니다.

깊이를 조정할 때와는 달리 컨버전스를 조정할 때는 화면 상에 컨버전스 레벨을 알려주는 그래픽 설명이 없습니다. 따라서 컨버전스를 변경할 때는 먼저 화면 상의 변경점을 주의깊에 확인하도록 합니다. 가장 좋은 방법은 오브젝트가 화면 또는 중간/레퍼런스 깊이에 있을 때 화면을 확인한 다음 이미지 이탈에 변화가 생길 때까지 (CTRL+F5)를 길게 누르는 것입니다. 컨버전스는 조금씩 증가하여 처음부터 확인하기 어렵기 때문에 이 과정에는 10 초에서 15 초 정도가 소요됩니다. 특수 안경 없이 볼 때 이미지 이탈을 없애려면 화면 깊이에 있는 오브젝트가 나타날 때까지 (CTRL+F5) 또는 (CTRL+F6)를 길게 누릅니다. 이렇게 하면 화면 깊이에 있는 오브젝트를 레퍼런스 지점으로 사용하여 컨버전스를 0 으로 설정합니다. 또, 선택된 컨버전스와 깊이의 수치가 정확하다고 판단되면, 해당 응용 프로그램을 나중에 사용할 수 있도록 (CTRL+F7)을 눌러서 레지스트리에 설정을 저장합니다.

NVIDIA 는 프로그램적으로 스테레오 3D 의 동작에 액세스하고 제어하는 API 및 스테레오 3D 를 사용하는 게임 기획에 대한 모범 사례가 정리된 문서를 제공합니다. 자세한 사항은 아래 링크를 참조해 주십시오.

API

http://developer.nvidia.com/object/nvapi.html.

Designing for stereo(스테레오를 위한 디자인)

http://developer.download.nvidia.com/presentations/2009/SIGGRAPH/3DVision_Develop_Design_Play_in_3D_Stereo.pdf.

6.2 Scaleform 스테레오 API

Scaleform 4.1 에서는 콘솔을 위해서 Render::HAL 내에 스테레오 3D 를 지원하는 간단한 API 를 도입하였습니다. 응용 프로그램에 따라서 스테레오 3D 에 맞춰 하드웨어를 초기화하고 프레임 버퍼 및 서페이스를 설정하여 HDMI 1.4 프레임 패킹을 지원할 수 있습니다.

섹션 6.2.1 에 나온 것처럼, 응용 프로그램은 처음에 Scaleform 에서 원하는 스테레오 파라미터를 초기화해야 합니다.

Scaleform 4 는 개별 스테레오 이미지를 나란히 또는 상하로 렌더링하도록 뷰포트를 설정할 수 있는 기능을 제공합니다. 이 기능은 스테레오 이미지를 바둑판 모양으로 세로 또는 가로로 정렬해야 하는 장치에 유용합니다.

이 작업은 해당 플래그로 **Render::Viewport::SetStereoViewport** 를 호출하여 수행할 수 있습니다.

다음은 SF4 에 새롭게 추가된 Viewport Stereo 옵션 플래그입니다.

```
// 디스플레이 하드웨어의 스테레오 전용(같은 크기의 버퍼를 사용하지만 양쪽 눈에는 절반씩 사용합니다.
View_Stereo_SplitV        = 0x40,
View_Stereo_SplitH        = 0x80,
View_Stereo_AnySplit    = 0xc0,

void SetStereoViewport(unsigned display);
```

6.2.1 Scaleform Stereo 초기화

```
Render::StereoParams s3DInfo;
s3DInfo.DisplayDiagInches = 46;      // 46 인치 TV
s3DInfo.DisplayAspectRatio = 16.f / 9.f;
s3DInfo.Distortion = .75;         // 0 에서 1 까지
s3DInfo.EyeSeparationCm = 6.4;    // 초기값인 6.4cm 로 합니다.
pRenderHAL->SetStereoParams(s3DInfo);
```

응용 프로그램은 디스플레이 순회 중에 양쪽 눈에 맞추서 Display 를 두 번 호출해야 하며, 각각의 Display 호출 전에 정확한 프레임 버퍼 서페이스로의 전환을 처리해야 합니다. Scaleform 는 양쪽 눈에 맞춰 뷰 오프셋을 처리합니다. 예제:

```
pMovie->Advance(delta);

pRenderer->GetHAL()->SetStereoDisplay(Render::StereoLeft);
pMovie->Display();

pRenderer->GetHAL()->SetStereoDisplay(Render::StereoRight);
pMovie->Display();
```

7 예제 파일

Scaleform SDK 에서 제공하는 3D 예제 Flash 파일(SWF, FLA 형식)을 확인하면 3Di 의 작동을 보다 확실하게 경험할 수 있습니다.

예제 파일의 저장 위치는 기본적으로 다음과 같습니다. *C:\Program Files (x86)\Scaleform\GFx SDK 4.4\Bin\Data\AS2 or AS3\Samples*

그림 4: 3DGenerator

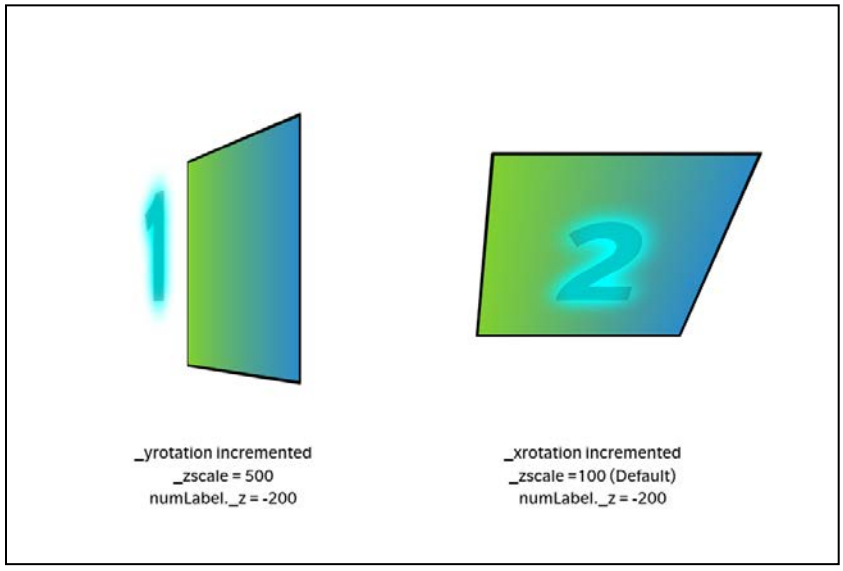

그림 5: 3D 사각형


3D Flash - tilted using ActionScript, no render textures used
ScaleformGFx
3D Window
Send
Enter Text
Change Image
Reset
Quit
Scaleform

그림 6: 3D 윈도우