

FrontPanelText 샘플

# *\*이 샘플은 Xbox One XDK와 호환됩니다 (2017 년 6 월)*

# 설명

FrontPanelText 샘플은 CPU를 사용하여 Xbox One X Devkit 전면 패널 디스플레이에서 텍스트를 그리는 방법을 보여줍니다. 이 샘플에서는 .rasterfont 파일을 로드할 수 있는 RasterFont라는 클래스를 사용합니다. 이름에서 알 수 있듯이 .rasterfont 파일에는 각 글리프가 래스터화된 간단한 픽셀 기반 글꼴이 들어 있습니다. 이 포맷은 CPU에서의 렌더링에 적합합니다. RasterFont 클래스는 텍스트를 쉽게 렌더링할 수 있는 printf-style 메서드를 제공합니다. PC에 설치된 True Type 글꼴에서 자신의 .rasterfont 파일을 만드는 방법에 대한 자세한 내용은 RasterFontGen 샘플을 참조하십시오.

# 샘플 사용하기

FrontPanelText 샘플은 전면 패널이 통합 된 Xbox One X DevKit 용입니다. 샘플을 시작하면 일부 샘플 텍스트가 전면 패널 디스플레이로 렌더링됩니다. 전면 패널 DPAD (왼쪽, 오른쪽)를 사용하여 텍스트의 글꼴 및 글꼴 크기 (위, 아래)를 변경하십시오. DPAD up을 사용하면 글꼴 크기가 커지고 DPAD down을 사용하면 글꼴 크기가 줄어 듭니다.



버튼당 1개의 LED

5x 프로그래밍 가능한 버튼

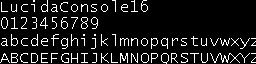
256 x 64 x 4bpp OLED 디스플레이

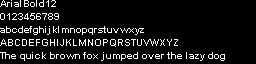
DPAD + 선택

DPAD 버튼을 눌러 (선택) 전면 패널 디스플레이에서 버퍼를 캡처하고 결과를 Title Scratch 폴더에있는 .dds 파일에 저장할 수 있습니다.

다음 이미지는 다양한 크기로 렌더링된 몇 가지 글꼴 옵션을 보여 주는 샘플의 스크린 샷입니다.

Lucida Console은 전면 패널에서 읽기가 가능합니다. 고정 너비 글꼴이며 텍스트에 예측 가능한 레이아웃 지오메트리가 있으므로 메뉴 및 위젯에 적합합니다. 또한, 여전히 12 픽셀의 높이에서 읽을 수 있으며 이 크기에서 5 줄의 텍스트가 디스플레이에 편안하게 맞습니다.

다음은 동일한 글꼴을 16 픽셀 크기로 렌더링 한 것입니다. 이 크기에서 디스플레이는 4 줄의 텍스트를 수용합니다.

Arial은 소문자 알파벳의 너비와 대문자 알파벳의 너비를 비교하여 알 수 있듯이 고정 너비가 아닙니다. Arial은 12 픽셀 높이에서 읽을 수 있으며 고정 너비 글꼴과 비교하여 디스플레이에 텍스트를 가로로 맞출 수 있습니다.

RasterFont 툴 체인을 사용하면 시스템에 설치된 TrueType 글꼴을 사용하여 .rasterfont 파일을 생성할 수 있습니다. 이것은 기호 글꼴의 예입니다. 기호 글꼴의 문자는 간단한 UI 요소 (예: 화살표, 단추 등)를 렌더링하는 데 사용할 수 있습니다 

# 구현 정보

프런트 패널의 텍스트는 RasterFont 객체를 사용하여 CPU에 렌더링됩니다. RasterFont 객체를 만들려면 .rasterfont 파일의 파일 이름을 생성자에 전달합니다. 예를 들면 다음과 같습니다.

auto myFont = RasterFont(L"Assets\\LucidaConsole16.rasterfont");

자신의 프로젝트의 경우 샘플과 함께 제공되는 .rasterfont 파일을 사용하거나 RasterFontGen.exe 도구를 사용하여 직접 만들 수 있습니다. RasterFontGen을 사용하면 시스템에 설치된 모든 True Type 글꼴에서 다양한 크기와 옵션으로 .rasterfont 파일을 만들 수 있습니다.

다음 코드 토막은 RasterFont 객체의 가상적인 엔드 투 엔드 (end-to-end) 사용법을 보여줍니다.

// .rasterfont 파일을 로드합니다.

auto myFont = RasterFont(L"Assets\\LucidaConsole16.rasterfont");

// 프런트 패널 디스플레이의 버퍼 디스크립터를 얻습니다.

BufferDesc fpDesc = m\_frontPanelDisplay->GetBufferDescriptor();

// 형식화된 문자열을 버퍼에 그립니다.

myFont.DrawStringFmt(fpDesc, 0, 0,

L"Simple Addition\n%i + %i = %i",

1, 1, (1 + 1));

// 프런트 패널에 버퍼를 표시합니다.

m\_frontPanelDisplay->Present();

BufferDesc는 CPU 버퍼의 너비와 높이를 추적하는 구조체입니다. RasterFont는 텍스트를 메모리의 임의의 주소로 렌더링할 수 있습니다. BufferDesc 버퍼의 크기를 기술해야 합니다. 프론트 패널 디스플레이의 대상을 보다 쉽게 하기 위해 샘플은 프론트 패널용 버퍼를 관리하는 FrontPanelDisplay 클래스를 사용합니다. RasterFont를 사용하여 프론트 패널에 텍스트를 렌더링 하는 데 적합한 BufferDesc를 얻기 위해 FrontPanelDisplay::GetBufferDescriptor()를 사용합니다.

DrawStringFmt는 텍스트를 버퍼에 그리는 데 사용됩니다. 이것은 표준 라이브러리 함수 printf()와 유사합니다. 참고로 BufferDesc외에 텍스트의 x 및 y 좌표가 요구됩니다. DrawStringFmt는 텍스트를 배치할 때 줄 바꿈을 지원합니다.

다음은 RasterFont에서 제공하는 텍스트 렌더링 방법에 대한 요약입니다.

// MeastureString 및 MeasureStringFMt는 계산에 유용합니다.

// 레이아웃 목적을위한 텍스트 범위

RECT MeasureString(const wchar\_t \*text) const;

RECT MeasureStringFmt(const wchar\_t \*format, ...) const;

// 버퍼에 대한 기본 텍스트 렌더링

void DrawString(const struct BufferDesc &destBuffer, unsigned x, unsigned y,

const wchar\_t \*text) const;

// 버퍼에 포맷 된 텍스트 렌더링

void DrawStringFmt(const struct BufferDesc &destBuffer, unsigned x, unsigned y,

const wchar\_t \*format, ...) const;

// 다음 DrawString 변형은 음영을 제공합니다.

// 다양한 회색 음영을 지정할 수 있는

// 매개 변수

void DrawString(const struct BufferDesc &destBuffer, unsigned x, unsigned y,

uint8\_t shade, const wchar\_t \*text) const;

void DrawStringFmt(const struct BufferDesc &destBuffer, unsigned x, unsigned y,

uint8\_t shade, const wchar\_t \*format, ...) const;

// 글리프 특정 메서드는 단일 글리프를 정확하게 배치하는 데 사용됩니다.

RECT MeasureGlyph(wchar\_t wch) const;

void DrawGlyph(const struct BufferDesc &destBuffer, unsigned x, unsigned y,

wchar\_t wch, uint8\_t shade = 0xFF) const;

RasterFont Notes:

* 그만큼 DrawString() shade 매개 변수가 있는 변형은 black-on-white 텍스트를 렌더링하는 데 사용할 수 있습니다. 예를 들어, 메뉴 시스템에서 선택된 행에 흑백 텍스트를 사용하려는 경우. 그 점에 유의하십시오. DrawString() 배경 픽셀을 그리지 않으므로 흰색 텍스트에 검은 색을 표시하려면 먼저 흰색 사각형을 그려야 합니다. 당신이 필요로 하는 직사각형의 범위를 결정하기 위해 MeasureString 메소드를 사용합니다.
* MeasureGlyph() 과 DrawGlyph()는 개별 글리프를 정확하게 배치하는 데 유용합니다. 이러한 메서드는 글리프의 테두리 상자만 사용하고 인접한 글리프 사이의 간격과 일반 텍스트 흐름을 배치하는 데 사용되는 세로 간격을 고려하지 않습니다. 이를 통해 글리프의 위치를 정확하게 지정할 수 있습니다. 예를 들어, 기호 글꼴의 글리프를 UI 요소 또는 위젯으로 사용하는 경우를 들 수 있습니다. (MeasureGlyph() 과 DrawGlyph() FrontPanelDemo 샘플에서 몇 가지 예를 찾을 수 있습니다.)
* RasterFont에서 제공하는 기본 텍스트 흐름에 만족하지 않으면, RasterGlyphSheet 클래스를 참조하세요. 이 클래스는 ForEachGlyph() 사용자 지정 텍스트 흐름 구현을 작성하는 데 사용할 수 있는 템플릿입니다. ForEachGlyph() 사용법에 대한 가장 좋은 예는 다양한 RasterFont::DrawString \*() 메소드 구현에서 찾을수 있습니다.

기타 구현 참고 사항:

* Xbox One X DevKit에서,::IsXboxFrontPanelAvailable() 은true값을 반환하고::GetDefaultXboxFrontPanel 는 유효한 IXboxFrontPanelControl 오브젝트를 반환합니다. 그렇지 않으면,::IsXboxFrontPanelAvailable() 는 false값을반환하고::GetDefaultXboxFrontPanel 은 nullptr를 반환합니다. (예 : Xbox One, Xbox One S 또는 실제 전면 패널이 없는 정품 콘솔)
* 매 프레임마다 전면 패널에 표시할 필요가 없습니다 (IXboxFrontPanelControl::PresentBuffer). 대신 하나 이상의 픽셀이 변경된 경우에만 표시해야 합니다. 따라서 샘플에서 m\_dirty 멤버는 디스플레이 버퍼가 변경 될 때마다 설정됩니다.
* 변경이 있을 때마다 조명 상태를 설정하기만 하면 됩니다.
* 전면 패널 버퍼에 직접 액세스할 수 없습니다. 대신 자신의 버퍼를 관리하고 버퍼의 주소를 IXboxFrontPanelControl::PresentBuffer에 전달해야 합니다. Sample::CaptureFrontPanelScreen() 는 간단히 DDS 표면에 대한 픽셀 페이로드로 m\_panelBuffer의 콘텐츠를 사용합니다.

# 업데이트 기록

2017 년 4 월 샘플의 첫 번째 릴리스.

# 개인정보처리방침

샘플을 컴파일하고 실행할 때 샘플 실행 파일의 이름이 Microsoft로 보내져 샘플 사용을 추적 할 수 있습니다. 이 데이터 수집을 거부하려면 Main.cpp에서 "샘플 사용 텔레메트리"라는 코드 블록을 제거하면 됩니다.

Microsoft의 개인 정보 취급 방침에 대한 일반적인 내용은 [Microsoft 개인 정보 취급 방침](https://privacy.microsoft.com/ko-kr/privacystatement/)을 참조하십시오.