게임프로그래밍

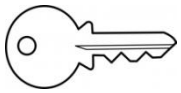
텍스트

박종승

Dept. of CSE, Incheon Nat. Univ.
jong@inu.ac.kr
http://ecl.inu.ac.kr

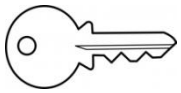
목차

- 텍스트 개요
- 텍스트 그리기 함수들
- 텍스트 그리기
- 고급 텍스트 그리기



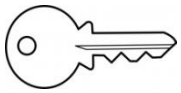
텍스트 개요

- 텍스트 처리
  - 레이아웃의 다양성으로, 처리가 매우 복잡함
  - 이를 위해서 DirectWrite API가 IDWriteTextLayout 인터페이스를 제공함
    - 다른 폰트들, 폰트 크기들,밑줄, 중간줄, 양방향 텍스트, 효과 등
    - DirectWrite API의 접두어: DWrite-, IDWrite-
- 텍스트 그리기
  - D2D는 텍스트 API인 DirectWrite API와 상호운용 가능함
  - 렌더타겟은 DirectWrite 텍스트 자원을 렌더링하는 함수들을 제공함 (DirectWrite의 쉬운 사용을 위함)
    - DrawText, DrawTextLayout, DrawGlyphRun
  - D2D는 텍스트의 렌더링을 위해서 GPU를 사용하므로, GDI를 사용하는 것보다 D2D를 사용하는 것이 CPU를 작게 사용함



텍스트 그리기 함수들

- ID2D1RenderTarget::DrawText
  - 간단한 텍스트 렌더링 방법을 제공함
    - 최소한의 formatting 사용
- ID2D1RenderTarget::DrawTextLayout
  - 더 복잡한 layout과 융통성을 제공함
  - IDWriteTextLayout 객체로 렌더링할 내용과 formatting을 명시함. 일부 문자열의 formatting도 명시할 수 있음.
- ID2D1RenderTarget::DrawGlyphRun
  - glyph-level의 정교한 제어가 가능함
  - DirectWrite의 measurement 지원함수들을 사용하여 제어값들을 얻음



텍스트 그리기

03.HelloWorld

- 텍스트를 단순하게 그리는 방법
  - 텍스트를 단일 포맷으로 그림
  - ID2D1RenderTarget::DrawText 함수
  - 참고: DirectWrite 관련 객체들은 장치 독립적임
    - IDWriteFactory, IDWriteTextFormat 객체 등
    - 효율성을 위해서: 응용의 초기화에서 한번만 생성

- 절차
  - 단계1: D2D 팩토리 및 DWrite 팩토리 생성

```
ID2D1Factory* m_pD2DFactory = NULL;
D2D1CreateFactory( D2D1_FACTORY_TYPE_SINGLE_THREADED, &m_pD2DFactory );

IDWriteFactory* m_pDWriteFactory = NULL;
DWriteCreateFactory( DWRITE_FACTORY_TYPE_SHARED,
                     __uuidof(m_pDWriteFactory),
                     reinterpret_cast<IUnknown **>(&m_pDWriteFactory) );
```

텍스트 그리기'

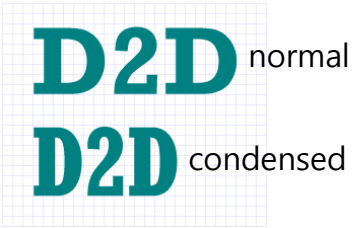
- 절차'
  - 단계2: IDWriteTextFormat 객체 생성
    - IDWriteFactory::CreateTextFormat 함수를 호출

```
IDWriteTextFormat* m_pTextFormat = NULL;
m_pDWriteFactory->CreateTextFormat( L"Verdana", NULL,
                DWRITE_FONT_WEIGHT_NORMAL, DWRITE_FONT_STYLE_NORMAL,
                DWRITE_FONT_STRETCH_NORMAL,
                50, L"", &m_pTextFormat );

m_pTextFormat->SetTextAlignment(DWRITE_TEXT_ALIGNMENT_CENTER);
m_pTextFormat->SetParagraphAlignment(DWRITE_PARAGRAPH_ALIGNMENT_CENTER);
```

텍스트 그리기'

- 절차"
  - 단계2: IDWriteTextFormat 객체 생성'
    - IDWriteFactory::CreateTextFormat 함수 인자들
      - 인자1: fontFamilyName; 문자열; 예: L"굴림체";
      - 인자2: fontCollection; NULL이면 시스템 폰트 컬렉션을 사용함을 의미
        - » IDWriteFontCollection 객체
      - 인자3: fontWeight;
        - » DWRITE_FONT_WEIGHT_*; *=THIN, EXTRA_LIGHT, ULTRA_LIGHT, LIGHT, NORMAL, REGULAR, MEDIUM, DEMI_BOLD, SEMI_BOLD, BOLD, EXTRA_BOLD, ULTRA_BOLD, BLACK, HEAVY, EXTRA_BLACK, ULTRA_BLACK
      - 인자4: fontStyle;
        - » DWRITE_FONT_STYLE_*; *=NORMAL, OBLIQUE, ITALIC

      - 인자5: fontStretch;
        - » DWRITE_FONT_STRETCH_*; *=ULTRA_CONDENSED, EXTRA_CONDENSED,CONDENSED, SEMI_CONDENSED, NORMAL, MEDIUM, SEMI_EXPANDED,EXPANDED, EXTRA_EXPANDED, ULTRA_EXPANDED
      - 인자6: fontSize; 장치독립픽셀(DIP) 단위; 1DIP=1/96inch;
        - » 픽셀단위 폰트크기임
      - 인자7: localeName: 예: "", "ko-kr";



텍스트 그리기'

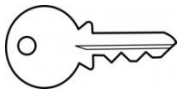
- 절차'''
  - 단계3: IDWriteTextFormat의 SetXXX() 함수 호출
    - IDWriteTextFormat::SetTextAlignment 함수 인자들
      - 인자1: textAlignment : 문장 내에서의 수평 정렬방법을 명시함
        - » DWRITE_TEXT_ALIGNMENT_*;*=LEADING,TRAILING,CENTER
        - » Leading/trailing/center는 문단의 왼쪽 모서리/오른쪽 지점/중심을 레이아웃 상자의 왼쪽/오른쪽/중심에 맞춤

    - IDWriteTextFormat::SetParagraphAlignment 함수 인자들
      - 인자1: paragraphAlignment : 문장 내에서의 수직 정렬방법을 명시함
        - » DWRITE_PARAGRAPH_ALIGNMENT_*; *=NEAR, FAR, CENTER
        - » near/far/center는 문단의 첫/마지막/중간 줄을 레이아웃 상자의 상단/하단/중간에 맞춤

텍스트 그리기”

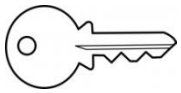
- 절차””
  - 단계4: DrawText를 호출하여 텍스트 그리기
    - ID2D1RenderTarget::DrawText 함수를 호출

```
D2D1_SIZE_F renderTargetSize = m_pRenderTarget->GetSize(); //렌더타겟의 크기 얻기

m_pRenderTarget->BeginDraw();
m_pRenderTarget->SetTransform(D2D1::Matrix3x2F::Identity());
m_pRenderTarget->Clear(D2D1::ColorF(D2D1::ColorF::White));

m_pRenderTarget->DrawText( L"Hello, World!", ARRAYSIZE(sc_helloWorld) - 1,
            m_pTextFormat,
            D2D1::RectF(0, 0, renderTargetSize.width, renderTargetSize.height),
            m_pBlackBrush );

m_pRenderTarget->EndDraw();
```



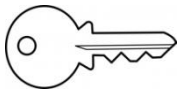
고급 텍스트 그리기

- 고급 텍스트를 그리는 방법
  - 복잡한 경우
    - 텍스트가 여러 포맷을 가진 경우 (예: 텍스트 일부를 밑줄)
    - 고급 OpenType 특성을 사용하는 경우
    - 충돌 테스트를 하는 경우
  - ID2D1RenderTarget::DrawTextLayout 함수를 사용
    - IDWriteTextLayout 객체를 렌더링함
    - IDWriteTextLayout 객체는:
      - IDWriteFactory::CreateTextLayout 를 호출하여 생성함
  - 더 자세한 내용: DirectWrite 파트 참조



텍스트 그리기에서 성능 측면

- 더 적절한 방법 선택하기
  - DrawTextLayout Vs. DrawText
    - 둘다 DirectWrite API로 formatting된 텍스트를 쉽게 렌더링할 수 있음
    - DrawTextLayout는 기존 IDWriteTextLayout 객체를 렌더링함
      - layout객체를 인자로 받으므로, 동일 텍스트가 여러 번 렌더링되는 경우에 적절함
        - » 한번만 IDWriteTextLayout 객체를 생성해둠
        - » 포메팅을 위한 측정 및 배치 작업은 객체 생성시에 한번만 수행함
        - » 텍스트를 그릴때마다 객체를 반복 사용함
    - DrawText는 인자값들을 사용하여 DirectWrite layout를 생성해줌.
      - 매번 호출시마다 layout을 생성함
  - Aliased Vs. AntiAliased 텍스트
    - 텍스트 품질이 최고가 아니어도 된다면 aliased 텍스트를 선택하자
      - 특별한 경우만 (텍스트 품질이 매우 중요한 경우만), antialiased로 하자



참고: 텍스트 렌더링과 관련된 추가 주제

- DirectWrite 개요
  - DirectWrite 사용법
    - 헤더파일: Dwrite.h
    - 링크파일: Dwrite.lib
    - DLL파일: Dwrite.dll
  - DirectWrite 안내서 (link: MSDN)

- Direct2D와 DirectWrite를 함께 사용하는 텍스트 렌더링
  - 두 요소의 상호운용의 장점
  - "Text Rendering with Direct2D and DirectWrite" (link: MSDN)