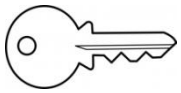
게임프로그래밍

자원

박종승

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



목차

- [참고: “렌더타겟”→this→“변환”]
- D3D 자원
- 장치 독립적 자원
- 장치 의존적 자원
- Factory 자원의 공유
- 렌더타겟 자원의 공유

D2D 자원

- D2D API의 최상위에 있는 인터페이스
  - ID2D1Factory
    - D2D 사용의 출발점
    - D2D 자원 객체를 생성함
  - ID2D1Resource
    - D2D 자원 객체
    - ID2D1Factory를 제외한 모든 객체들은 이 인터페이스를 상속함

- 두 가지 타입의 D2D 자원
  - 장치 독립적 자원: CPU에서 관리
  - 장치 의존적 자원: GPU의 자원과 직접 매핑됨

장치 독립적 자원

- 장치에 독립적인 자원들
  - 항상 CPU쪽에 상주함. 렌더링 하드웨어와 연관되어 있지 않음.
  - 응용의 수명 동안 존재할 수 있음

- 장치 독립적 자원들: 종류 (이것들만!!)
  - ID2D1Factory
  - ID2D1Geometry 및 이를 상속한 인터페이스들
  - ID2D1GeometrySink 및 ID2D1SimplifiedGeometrySink
  - ID2D1StrokeStyle
  - ID2D1DrawingStateBlock

- 장치 독립적 자원들: 생성하는 방법
  - ID2D1Factory : CreateFactory 함수로 생성함
  - 그 외 : ID2D1Factory의 함수들을 사용하여 생성함
  - 참고: ID2D1Factory는 렌더타겟들을 생성함 (장치 의존적 자원임)
    - 렌더타겟 이외의 ID2D1Factory가 생성하는 모든 자원들은 장치 독립적 자원임!!

장치 의존적 자원

- 장치 의존적 자원
  - 특정한 렌더링 장치와 연관되어 있음
    - 장치?: 하드웨어 가속이 가능한 경우 "GPU", 그 외의 경우는 "CPU"
  - 해당 장치가 제거되면 자원도 기능이 정지됨
  - 언급된 장치 독립적 자원들 이외의 모든 자원들이 이에 해당함

- 장치 의존적 자원의 생성
  - 렌더타겟의 함수를 호출하여 생성함
  - 렌더타겟 자체는 factory의 함수를 호출하여 생성됨

- 장치 의존적 자원의 예
  - ID2D1RenderTarget와 이를 상속한 것들
  - ID2D1Brush와 이를 상속한 것들
  - ID2D1Layer

- 렌더타겟이 가용하지 않게 되면
  - 연관된 모든 장치 의존적 자원들도 사용할 수 없게 됨
  - 렌더타겟이 소실되어 D2DERR_RECREATE_TARGET 에러가 발생되면,
    - 렌더타겟을 재생성
    - 모든 관련 자원도 재생성

ID2D1Factory 인터페이스

- ID2D1Factory의 생성
  - D2D1CreateFactory() 함수를 호출하여 생성함
  - CreateXXX 함수들을 제공함
- ID2D1Factory가 CreateXXX로 생성하는 자원
  - 렌더타겟: CreateXXXRenderTarget
    - 그리기 명령들을 수행하는 객체
  - 그리기 상태 블록: CreateDrawingStateBlock
    - 그리기 상태 정보를 저장하는 객체
      - 상태정보의 예: 현재의 변환, anti-aliasing 모드
  - 기하: CreateXXXGeometry
    - 모양을 표현하는 객체

Factory 자원의 공유

- Factory가 생성한 모든 장치 독립 자원들
  - 동일 factory가 생성한 모든 다른 자원들과 공유할 수 있음(장치 독립적이든 장치 의존적이든)
  - 예: 한 factory로 두 ID2D1RenderTarget 객체를 생성한 경우,
    - 두 렌더타겟 모두 한 동일 ID2D1RectangleGeometry 를 그릴 수 있음
  - 참고: sink 인터페이스 (ID2D1SimplifiedGeometrySink, ID2D1GeometrySink, ID2D1TessellationSink)는 어떤 factory가 생성한 자원과도 공유될 수 있음

렌더타겟 자원의 공유

- 렌더타겟 자원의 공유
  - 한 렌더타겟이 생성한 자원을 공유할 수 있는지의 여부
    - 렌더타겟의 종류에 따라 다름
  - 렌더타겟을 D2D1_RENDER_TARGET_TYPE_DEFAULT로 생성한 경우,
    - 이 렌더타겟이 생성한 자원은 그 렌더타겟만 사용할 수 있음
      - 예외조항이 있음 (다음 쪽 참조)

타 렌더타겟 자원의 공유

- 한 렌더타겟이 생성한 자원을 다른 렌더타겟이 공유할 수 있는 경우
- 각 렌더타겟에 따라서 다름
  - 하드웨어 렌더타겟
  - 소프트웨어 렌더타겟
  - "DXGI 표면 렌더타겟"
  - "Compatible 렌더타겟"
  - "공유된 비트맵"

타 렌더타겟 자원의 공유’

- 하드웨어 렌더타겟이 생성한 자원의 공유 여부
  - 명시적으로 하드웨어를 사용하는 모든 다른 렌더타겟과 공유할 수 있음
    - 단, 원격 모드가 compatible해야 함
  - 원격 모드가 compatible 하려면, 두 렌더타겟 모두 usage 플래그 명시 방법이 다음 중 하나에 해당하는 경우임
    - D2D1_RENDER_TARGET_USAGE_FORCE_BITMAP_REMOTING, 또는
    - D2D1_RENDER_TARGET_USAGE_GDI_COMPATIBLE, 또는
    - 두 플래그 모두 명시되지 않은 경우
  - 위의 세팅에 해당하는 경우는, 자원이 항상 동일 컴퓨터에 있는 경우임.
  - 렌더타겟의 생성 시에 속성 명시 방법
    - D2D1_RENDER_TARGET_PROPERTIES 구조체의 usage 필드, type 필드 등에 명시
  - 하드웨어를 명시적으로 사용하는 렌더타겟을 생성하는 방법
    - type 필드에 D2D1_RENDER_TARGET_TYPE_HARDWARE를 명시
- 소프트웨어 렌더타겟이 생성한 자원의 공유 여부
  - 명시적으로 소프트웨어 렌더링을 사용하는 모든 다른 렌더타겟과 공유할 수 있음
  - 소프트웨어 렌더링을 명시적으로 사용하는 렌더타겟을 생성하는 방법:
    - type 필드에 D2D1_RENDER_TARGET_TYPE_SOFTWARE를 명시
  - 참고: “WIC 비트맵 렌더타겟”은 항상 소프트웨어 렌더타겟으로 간주됨

타 렌더타겟 자원의 공유”

- “DXGI 표면 렌더타겟”이 생성한 자원
  - 동일 D3D 장치를 사용하는 모든 다른 DXGI 표면 렌더타겟과 자원을 공유할 수 있음
- “Compatible 렌더타겟”
  - 한 렌더타겟과 그 렌더타겟이 생성한 “Compatible 렌더타겟”들 간에는 자원을 공유할 수 있음
  - “Compatible 렌더타겟”을 생성하는 방법:
    - ID2D1RenderTarget::CreateCompatibleRenderTarget 함수를 호출
- “공유된 비트맵”
  - ID2D1RenderTarget::CreateSharedBitmap 함수로 두 렌더타겟이 공유할 수 있는 비트맵(ID2D1Bitmap)을 생성할 수 있음
  - 두 렌더타겟이 동일 장치를 사용하면 함수가 성공 수행됨

효율적인 자원의 사용

- 자원의 재사용
  - 자원의 생성 및 소멸 주체: 소프트웨어 또는 하드웨어
  - 하드웨어가 수행하는 경우는 비용이 비쌈
    - 이유: 비디오카드와의 과도한 통신이 필요하므로
  - 따라서 가급적 소프트웨어적으로 자원을 재사용해야 함.
  - 예: 게임 장면을 구성하기 위한 비트맵 자원들을 응용 시작 시점에서 모두 생성함.
    - 렌더링 시에 필요한 비트맵들을 재사용함 (재생성 하지 않음)
  - 예외: 윈도우 resize 연산의 경우, 스케일에 의존된 자원들은 모두 재생성