엔진을 스레드로 구동시키기2 [2016.08.18]

Sample 개요

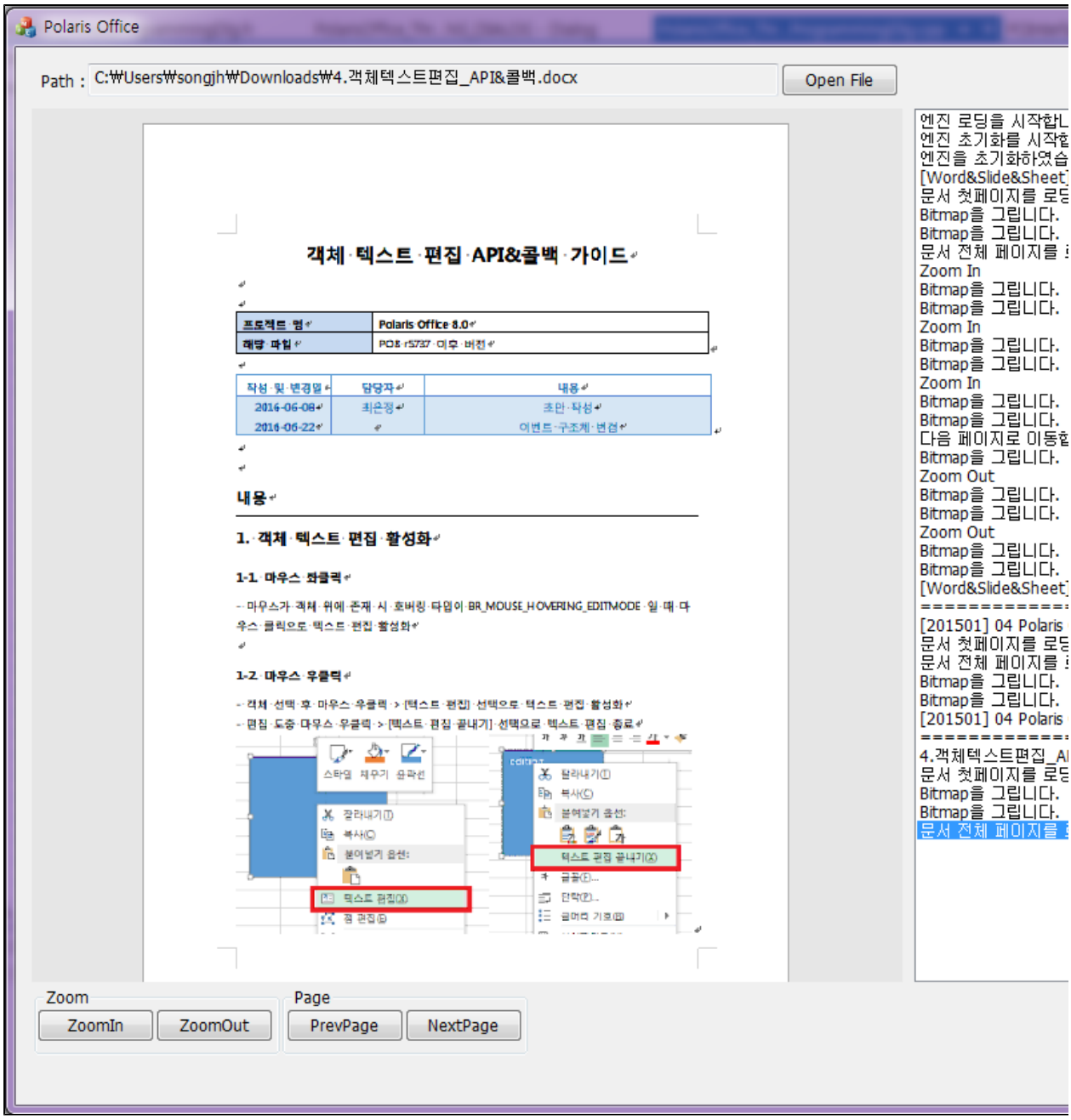
a. 기존 PolarisOffice 제품에서는 하나의 스레드에서 엔진, UI를 구동시켰다. 이번 과제를 통해 UI와 **별도의 스레드에서 Engine을 구동**하도록 한다.
b. 사용하는 스레드는 UIThread이며 메세지큐를 활용하여 엔진을 구동시킨다.

Sample svn

a. https://tr00129.infraware.net/svn/PolarisOffice7_Engine/POTester/ThreadSample
b. 시작 프로젝트 : PolarisOffice_ThreadProgramming
c. **EngineAPI, EngineDLL이 빌드되어 있어야 한다. (각 프로젝트의 결과물이 있어야 실행됨.)**

Sampe UI

1. UI 구성



a. File Path 설정 버튼 / 문서 비트맵 출력창 / 상태 로그 출력창
b. Zoom In / Zoom Out / Prev Page / Next Page

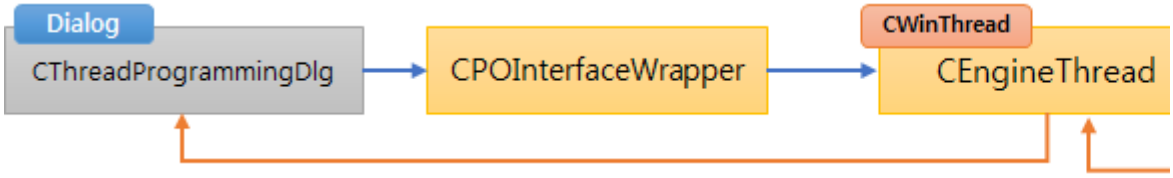
추가 구현 항목

1. **엔진을 PO8로 교체**
    a. Saas PC 의 엔진 인터페이스를 사용하여 교체한다.
    b. Saas PC Engine 과 CPOInterface의 연결이 필요
2. **문서 오픈 시 화면에 비트맵 출력**
    a. Picture Control을 추가하여 문서 비트맵을 출력한다.
3. **간단한 Viewer 기능 연결**
    a. Zoom In/Out
    b. Prev page, next page

구현 항목 세부 사항

1. 엔진을 PO8로 교체

a. 엔진 호출 구조



b. Engine
   i. Saas PC Client의 EngineAPI, EngineDLL, PInterface 로 교체
c. CPOInterface
   i. EngineAPI 와 연결하기 위해 IEngineListener를 상속받음

2. 비트맵 출력

a. 비트맵 출력 관련 callback 함수 구현

구현 시 주요 이슈

1. CEngineThread에서의 다중 상속

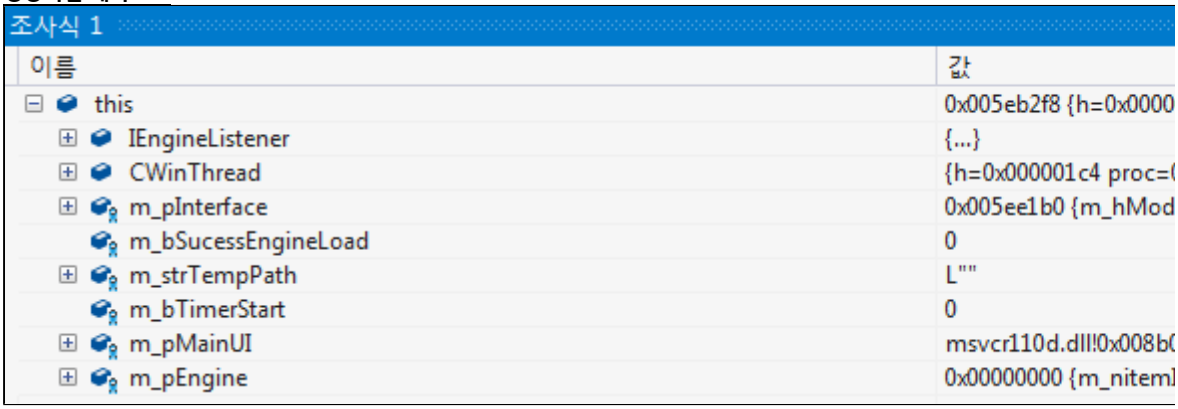
a. EngineCallback Listener를 연결하기 위해 CWinThread에서 아래와 같이 다중 상속을 받는 경우 메모리가 깨지는 문제가 발생

```
class CEngineThread : public IEngineListener, public CWinThread
```

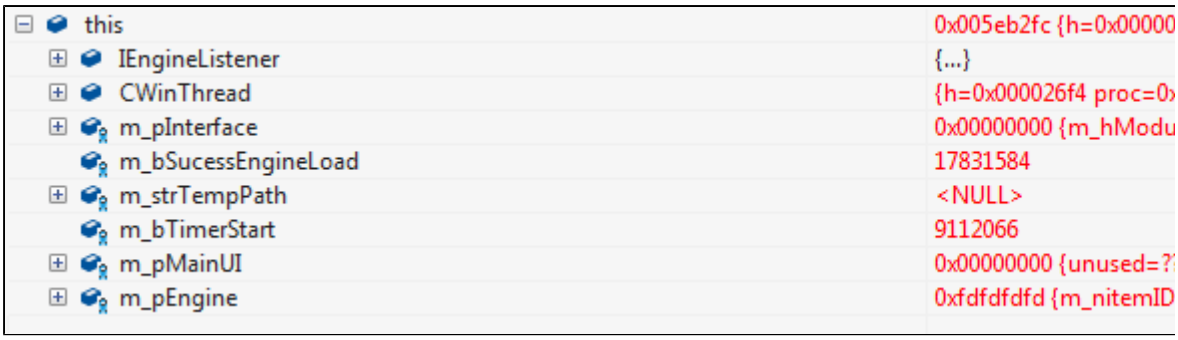
b. PostMessage가 전달된 함수 내부에서 this가 깨져있다.

```
void CEngineThread::ThreadManager(WPARAM wParam, LPARAM lParam)
{
 if(wParam == NULL) return;
 BaseEvent* pEvent = (BaseEvent*)wParam;
 switch(pEvent->nType){
 case ENGINE_LOAD:
  break;
 case ENGINE_INITIALIZE_EVENT:
  break;
 case ENGINE_OPEN_DOCUMENT_EVENT:
  break;
 case ENGINE_TIMER:
  break;
 case ENGINE_CLOSE_DOCUMENT_EVENT:
  break;
 }
 delete pEvent;
}
```

정상적일 때의 this



다중상속 받았을 때의 this



c. 해당 문제 관련하여 구글링한 결과 (링크 참고)
   i. The message handling logic makes the assumption that there is only one base class when passing the message up the line. Multiple inheritance is a no-no.
   ii. http://www.itlisting.org/5-windows/728a4339bbbb08bd.aspx
d. 해결책 : UIThread에서 다중상속을 받지 않고, POInterface에서 engine callback listener 를 상속받는다.

2. 비트맵 동기화 문제

a. 동기화가 필요한 이유
   i. 엔진과 UI는 각각의 Thread에서 동작하며 Bimap buffer를 공유하고 있다.
   ii. 엔진 - Bitmap buffer write / UI - Bitmap buffer read
   iii. 각 Thread가 Bitmap buffer에 동시에 접근하지 못하도록 동기화 과정이 필요하다.

b. 동기화 과정 추가 - Critical Section 사용

POInterface.cpp

```
void CPOInterface::OnLockUIBuffer(int **a_pBuffer, int a_nWidth, int a_nHeight)
{
 g_CriticalSection_For_Draw.Lock();
 MakeBitmap(a_nWidth, a_nHeight, 32);
 *a_pBuffer = (int*)(GETDATAPOS(m_pDIB));
}
```

OnDrawBitmap() 호출 시 bDrawEvent를 TRUE로 세팅. TRUE인 경우 OnLockUIBuffer() 함수 내부에서 Lock 발생

POInterface.cpp

```
void CPOInterface::OnUnLockUIBuffer()
{
 g_CriticalSection_For_Draw.Unlock();
}
```

UI가 Paint를 끝낸 후 Lock을 해제

PolarisOffice_ThreadProgrammingDlg.cpp

```
void CPolarisOffice_ThreadProgrammingDlg::OnPaint()
{
 // 엔진에서 전달받은 bimap을 그린다.
 if(bDrawEvent)
 {
  g_CriticalSection_For_Draw.Lock();
  if(m_pBitmap)
  {
   //CWnd *parent = GetParent();
   CDC *pDC = m_picture.GetDC();
   CRect rc;
   m_picture.GetWindowRect(&rc);
   ::StretchDIBits(pDC->GetSafeHdc(),  // handle to device context
    0,    // x-coordinate of upper-left corner of dest. rectangle
    0,                // y-coordinate of upper-left corner of dest. rectangle
    rc.Width(),          // width of destination rectangle
    rc.Height(),        // height of destination rectangle
    0,               // x-coordinate of upper-left corner of source rectangle
    m_pBitmap->biHeight+1,               // y-coordinate of upper-left corner of source rectangle
    m_pBitmap->biWidth,         // width of source rectangle
    -m_pBitmap->biHeight,       // height of source rectangle
    GETDATAPOS((LPBITMAPINFOHEADER)m_pBitmap), //GETDATAPOS(m_pBitmap),          //
address of bitmap bits
    (LPBITMAPINFO)m_pBitmap,//LPBITMAPINFO address of bitmap data
    DIB_RGB_COLORS, // usage flags
    SRCCOPY);// raster operation code
  }
  g_CriticalSection_For_Draw.Unlock();
 }
}
```