모바일 플랫폼상의 X-File Viewer

(X-File Viewer on a Mobile Platform)

하 경 주*
(Kyeoung Ju Ha)

요 약 본 논문에서는 다양한 모바일 플랫폼 환경에서 실행이 가능한 모바일 문서 뷰어를 제안한다. 제안한 문서 뷰어는 자체 파일분석을 통한 파일 decoding 엔진과 독립적인 모듈로 동작하는 엔진으로 OS에 따라 유연하게 장착이 가능하게 설계되었다. 또한 제안된 뷰어에서는 문서파일의 특징을 분석하여 문서파일의 editing tool들의 기초 자료로 사용되어질 수 있다.

핵심주제어 : X-File 뷰어, 문서 뷰어, 모바일 문서 뷰어, 디코딩 엔진

Abstract In this paper, we propose a mobile document viewer which is executable on a variety of mobile platforms. The proposed viewer is designed with file decoding engine and independent module which are adopted with os independent. The proposed viewer can be used as a basis of document editing tool by analyzing the characteristics of the document file.

Key Words : X-File viewer, document viewer, mobile document viewer, decoding engine

1. 서 론

21세기는 정보화 사회로 정보기술을 중심으로 다양한 통신망 환경에서 다양한 형태의 정보 서비스가 이루어지는 사회구조를 형성하게 되어 있으며, 특히 무선 통신 기술을 이용함으로써 업무나 개인적 용무의 경우 장소와 상관없이 다양한 형태의 정보 서비스가 이루어지고 있다.

또한 무선통신의 급성장 추세는 지속될 것으로 보이고, 이러한 추세에 따라 각 통신 사업자마다 환경변화에 대응하고 경쟁력을 높이기 위해 다양한 형태의 정보를 제공하는 사업으로 발전적인 변화가 예상되며, 이에 따른 데이터 통신이 시장에서 급성장할 것으로 전망되고 있다.

통신기기 및 데이터 통신 환경의 발전으로 인하여, 언제 어디서나 e-Business가 가능해지고 있으며, 다양한 e-Business를 위한 솔루션의 수요가 늘어나고 있다. 이에 따라 모바일 기기의 이동성과 결합된 모바일 문서뷰어 시장이 주목받고 있다. 모바일 문서뷰어란 모바일기기에서 다양한 형태의 문서와, E-Mail, HTML, Photo 파일 등을 view, search, management 하는 기능이 결합된 솔루션을 말하고 있으며, 과거 모바일 기기의 협소한 플랫폼 환경을 극복하고 현재 다양한 서비스를 이용할 수 있는 플랫폼이 등장함에 따라, 수요의 계층도 다양해지고 있다.

휴대폰의 인터넷과 메일기능이 보편화 된 시점에서 메일에 첨부 파일 뿐만 아니라 웹상의 수많은 문서를 휴대폰으로 쉽게 활용할 수 있다는 점에서 모바일 문서뷰어 서비스가 보편화 될 것으로 예상되고 있으며, 휴대폰 디스플레이 환경이 발전하면서 활용분야도 한층 넓어질 것으로 전망하고 있다.

이동통신 사업자들의 모바일 문서뷰어 서비스가 구

* 대구한의대학교 모바일콘텐츠학부

체화됨에 따라 솔루션 선정 작업이 이루어지고 있는 시점에서, 국내 모바일 문서뷰어 시장은 외국 솔루션의 의존도가 높고, 기존의 국내 솔루션은 사용의 불편함이 야기되고 있는 실정이다. 따라서 외국 솔루션의 다양하고, 편리한 기능에 부합되는 국내 솔루션의 개발에 의견이 모아진다.

본 논문에서는 다양한 모바일 플랫폼 환경에서 실행이 가능하고, 자체 파일분석을 통한 파일 디코딩(decoding) 엔진개발과 함께, 사용성을 고려한 설계에 초점을 둔, 모바일 문서뷰어 시스템을 제안한다.

모바일 문서뷰어의 핵심은 여러 문서를 해석할 수 있느냐 인데, X-File viewer에서는 문서파일의 특징을 분석하여, 자체 decoding 엔진을 구현하며, 이는 문서 파일들의 editing tool들의 기초 자료로 사용되어질 수 있다.

또한 현재 내수 시장에 사용되고 있는 모바일 문서 뷰어의 경우 외국 솔루션의 의존도가 높은데, 국산화를 통하여 국내 통신회사, 제조업체의 marketing power를 통한 내수 및 해외 시장의 판로를 개척할 수 있으며, 기존 이동통신 단말기들에 비해 향상된 문서 뷰어 서비스를 갖춘 단말기가 출시됨에 따라 국제 단말기 시장에 변화가 예상되며, 이에 따른 지역 산업의 혁신을 주도할 수 있는 기틀을 마련할 것으로 기대되고 있다.

유무선 연동의 확대와 함께 각종 문서를 휴대폰에서 확인하거나 활용하는 것이 보편적 서비스로 자리잡을 것으로 예상되며, 문서 뷰어의 활성화가 인터넷과 연결된 각종 응용 서비스 개발로 이어질 수 있어서 이동통신 사업자의 서비스 확대를 기대되고 있다. 커다란 성장 잠재력을 가지고 있는 모바일시장의 성장을 위해서는 관련 기술의 개발 및 주도권을 잡기 위한 조기 연구가 시급한 상황이며, 개발 완료시 업체들의 치열 한 경쟁이 예상됨으로 기술력 확보로 인한 시장 선점의 효과가 아주 클 것으로 기대되고 있다.

2. 국내외 관련 연구

국내 솔루션 전문 업체 '드림투리얼리티(이하D2R社)'에서 multi file viewer 프로그램인 CSD viewer를 개발하였으며, 'CSD'는 D2R社와 ㈜한글과컴퓨터(이하 한컴社)가 공동으로 개발한 viewer와 font를 포함한 전자 문서 통합 포맷이며 2004년 한국 신기술(NT) 인증(산업자원부 기술표준원에서 선정)을 획득하였으며, CSD포맷으로 변환된 파일을 폰에 Down받아서 폰에서 보는 기술로 CSD포맷으로 변환해야 하는 선행 작업이 있어야 하는 번거로움이 있다.

차세대 휴대폰에서는 메일에 첨부 파일 뿐만 아니라 웹상의 수많은 문서를 휴대폰으로 손쉽게 활용 하는 등의 문서 뷰어 서비스의 보편화가 예상되고 있다.

국외의 대표적인 기술 개발 업체로는 영국의 Picsel Technologies Ltd.(이하 Picsel社)가 있다. Picsel사는 mobile 환경에 맞는 multimedia content engine (ePAGE)을 개발하여 모든 형태의 전자문서로의 다양한 file format(HTML, MS Word, PDF, Flash, JPEG, GIF)의 문서에 접근하여 단말기 상에서 깨짐 현상 없이 display되도록 지원하는 App 솔루션인 PICSEL FILE VIEWER등을 구축하여 세계적인 기술로 인정받고 있다. Picsel사의 File viewer는 국내외 유명 휴대폰 단말기 제조업체(Intel, Motorola, Nokia, Samsung, Panasonic, Sharp, Sony)이외에도, 국제적으로 여러 제조업체(Bandai, Casio, Hitachi, KDDI, Khazana, Kyocera, NTT Do Co Mo, Palm, Qualcomm, TCL, ZTE)와 계약을 맺고, Picsel사의 App솔루션을 제공하고 있으며 여러 가지 확장 기술을 꾸준히 개발하여 세계적인 브랜드로 자리매김하고 있다.

앞서 기술한 바와 같이 국내에서 자체 엔진을 포함하는 뷰어는 전무한 상태여서 본 연구에서 제안하는 모바일 전자문서 뷰어기술은 향후 모바일 시장에서의 국산화 및 파급효과가 클 것으로 예상 된다.

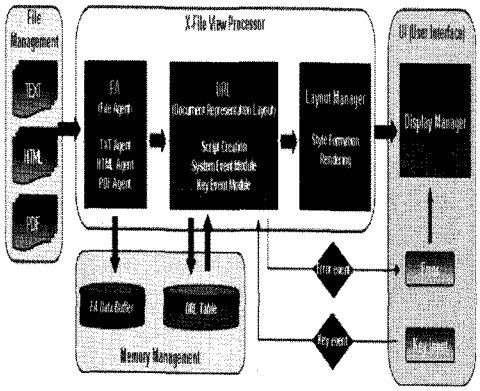
'X-File viewer'는 PC로 작성한 문서를 PC 및 주변기기 활용에서 벗어나 모바일 환경에서 문서 viewer가 가능하게 하여 시간과 장소에 구애 받지 않은 보다 편리한 정보 습득 및 활용이 가능하며, 모바일 전자문서 뷰어 시스템은 모바일 단말기의 기능을 확장하므로 모바일 사용자들의 욕구를 만족시킬 수 있을 것으로 기대된다.

3. X-File Viewer

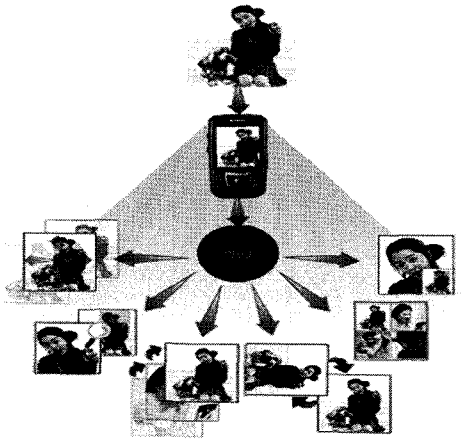
X-File viewer의 구성은 크게 세 가지의 document representation 모듈로 나누어져 있다. 첫 번째는

virtual layout 및 디스플레이를 위한 모듈인 plain TXT agent 엔진이며, 두 번째는 , HTML 4.01을 지원하기 위한 모듈이다. 그리고 마지막으로 PDF 1.4 format을 위한 virtual layout 및 디스플레이를 위한 모듈이다.

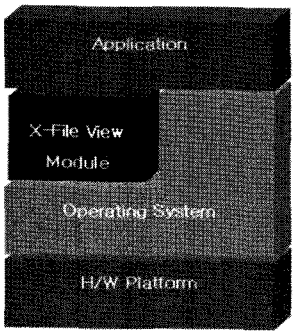
X-File viewer의 시스템 구성 및 기능은 <그림 1>과 같다.



(a) 시스템 구성



(b) X-File viewer의 기능



(c) X-File viewer architecture

<그림 1> X-File viewer의 시스템 구성 및 기능

X-File Viewer의 기능은 up, down, left, right로 화면 이동이 가능하며, 확대/축소(zoom in/out), 페이지 이동(page move), 회전(rotate), 전체화면/보통화면 보기가 가능하다.

또한, 파일 타입에 따라서 각각의 agent에서 구성 요소 별로 분석을 수행하고, 분석된 구성요소는 모두 동일한 format을 가지고 화면을 재구성한다. 재구성하여 Object화된 자료를 이용하여 화면 layout을 구성하여 화면을 출력한다. 그리고, Layout, Display 작업은 파일 타입에 관계없이 동일하게 동작한다.

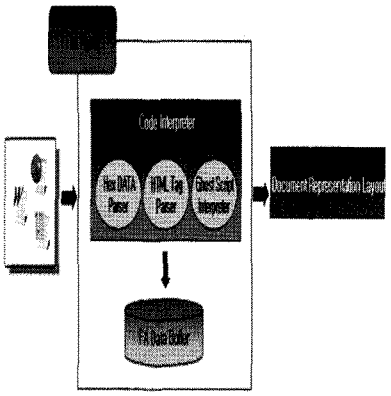
1) File agent

File agent는 파일 타입별 agent 엔진 개발 파일 분석, 스펙을 통한 타입별 유형별 분석 및 레이아웃, 문자, 표등 각각의 정보를 객체화 한다. 이러한 file agent의 역할은 <그림 2>와 같다.

첫 번째, PDF agent는 PDF 1.4까지 지원하고 다국어를 지원하며, POST-Script 언어 기반의 기술로 완성된 포맷이다. 이는 Gost-Script Interpreter를 사용하여 멀티미디어 오브젝트를 제외한 레이아웃, 폰트, 텍스트, 도표정보를 객체화한다.

두 번째, TXT agent는 ASCII, UTF8, Unicode, ANSI, 다국어 코드를 지원하며, 각 코드 별 문서데이터를 분리 객체화 한다.

세 번째, HTML agent는 HTML 4.0까지 지원하며 HTML tag parser를 이용하여 레이아웃, 문서 데이터 그리고 도표 정보를 객체화한다.



<그림 2> File agent의 역할

Document loading이 되면 file type을 구분 하여 해당 file agent를 호출하게 된다. 호출된 file agent는 원본 document를 한 페이지 단위로 loading 한 후 size를 계산하여 한 페이지에 대한 data값을 읽어 들인다. 읽혀진 data는 file agent 코드 해석기를 통해 불필요한 정보를 제거 후 실제 출력할 data code만 객체화 하여 임시 버퍼에 저장한다.

File agent 코드 해석기는 <그림 3>과 같이 한 페이지에 해당하는 data를 x_save_dataa() 함수를 호출하여 임시 버퍼에 저장 후 DRL 연동을 위해 xfv_drl_active()함수를 호출한다. 원본 document의 다음 페이지에 대한 data 값을 위와 동일하게 처리 한다.

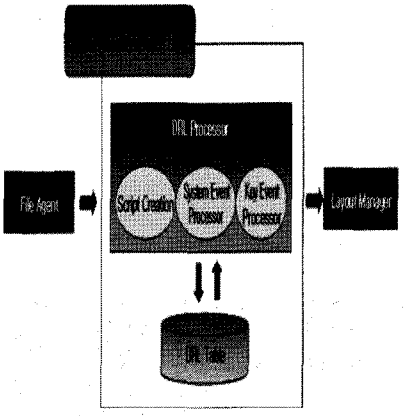
```
void Xfv_interpretation(int filetype, char *filename)
{
    /* loading files */
    xfv_convert(x_strBuffer, x_length);
    ...
    for(x_strBuffer != Null)
    {
        if(cmp_code != x_str) {
            xfv_delete_str(&x_str_Buffer);
            ...
        }
        else
        {
            x_save_data(&x_str_Buffer);
        }
    }
    ...
    return 0;
}
```

<그림 3> File Agent 코드 해석기

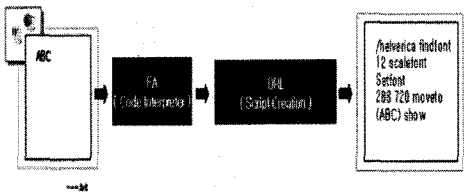
2) DRL

DRL(Document Representation Layer) agent는 <그림 4>와 같으며, 객체 정보를 바탕으로 문서를 재구성하며, DRL module에서 문서의 폰트가 설정된다. PDF 파일의 경우 내부폰트가 없을시 TrueType font로 대체하며, 그 이외의 문서는 기본적으로 내장될 기본 TrueType font를 사용하여 문서 전체를 벡터화하고, 문자와 표등을 벡터화 함으로써 페이지의 확대, 축소에서도 깨지지 않는 깨끗한 화면을 제공한다.

DRL은 file agent의 객체 정보를 바탕으로 문서를 재구성 역할 및 event 처리하며, DRL이 실제로 수행하는 일은 변환된 data값을 script 타입으로 재구성하여 실제 출력을 위한 정보들을 담고 있는 internal representation 을 생성한다. Internal representation은 <그림 5>와 같다.



<그림 4> Document representation layer module 구성



<그림 5> Internal representation 구성

실제 디스플레이를 위해 텍스트의 경우 글꼴, 글꼴 크기, 텍스트가 출력되는 위치정보 등이 필요하고, 출력을 위한 모든 오브젝트(텍스트, 이미지 등)는 문서상의 어느 곳에 위치하는지에 대한 좌표 값이 필수적으로 요구된다. DRL은 문서상의 여러 정보들을 객체화 시키고, 이 객체들은 화면상에 실제로 출력되기 위한 여러 정보들을 포함한다.

File agent에서 xfv_drl_active() 함수가 호출 되면 DRL은 file agent의 임시 Bbuffer를 검색 하고 script 타입으로 DRL table을 구성한다. DRL table은 file agent의 data 정보를 바탕으로 실제 출력될 data code와 size, position, page 정보 등을 script로 구성한다.

File agent의 임시 버퍼의 data가 존재 하지 않을 때까지 검색하여 file agent의 임시버퍼의 모든 data가 table로 구성이 되면 layout module을 사용하기 위해 xfv_layout()함수를 호출 한다. 이후 사용자가 확대, 회전 및 페이지 이동의 event를 발생시키면 DRL 모듈에서 위치, 사이즈, 페이지에 대한 정보를 바탕으로 <그림 6>과 같이 DRL table의 정보를 재 정의를 한다.

Event는 system에서 호출되는 system event와 외부적 발생의 interrupt의 key event로 나누어진다. Key event의 경우 X-File viewer에서 동작할 수 있도록 key map 형식으로 구성되어 key 정보에 대한 system event는 key event로 인한 error 리턴 호출 및 내부인 event를 관리한다. 이를 관리하는 event module은 <그림 7>과 같다.

```
void Xfv_drl_activet(Xfv_file_att file_att)
{
   if(xfv_tmp_data != NULL) {
      pagesize = context->size;
      ...
      xfv_tb_data = file_att->xname;
      xfv_tb_data = file_att->length;
      ...
   }
   return;
}
```

<그림 6> DRL 테이블 구성

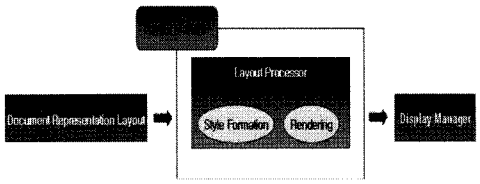
```
int XfvApp_keyPress(Xfv_Context *xfvContext,
                    XfvKeyCode keyCode,
                    int repeats);
int XfvApp_keyRelease(Xfv_Context *xfvContext,
                      XfvKeyCode keyCode);
int XfvApp_PointerDown(Xfv_Context *xfvContext,
                       unsigned int timestamp,
                       int x, int y);
int XfvApp_PointerMove(Xfv_Context *xfvContext,
                       unsigned int timestamp,
                       int x, int y);
```

<그림 7> Event module 구성

3) Layout manager

모바일 환경에 적합한 layout manager를 구성하며, 제약된 Display 환경에서 최적화된 layout, 화면에서의 자유로운 확대/축소, rotate 기능 지원, 멀티페이지 구성에 따른 페이지 이동 기능 구성을 제공한다. Layout module은 <그림 8>과 같다. 그리고 DRL에서 처리되어 생성된 DRL table을 이용하여 실제 모바일 단말기의 화면 구성에 맞게 구성하여 원본 document에 맞게 display하도록 style를 구성하고, 다양한 효과의 rendering처리와 layout 구성 기능을 수행한다. 이 모듈의 구성은 <그림9, 10>과 같다.

DRL table의 정보를 바탕으로 모바일단말기에서 출력될 실제 사이즈에 맞도록 style을 구성한 후 display 할 data의 다양한 효과를 rendering처리 하여 target display에 적합한 객체로 생성 구성한다.



<그림 8> Layout manager의 구성

```
void Xfv_layout_formation()
{
   if(xfv_tb_data)
   {
      ...
      x_ly = xfv_tb_data -> page_info;
      ...
   }
   xfv_ly_SetStatus();
   XFV_CFG_info = xfv_getScreen_info();
   Xfv_Render_set(XFV_CFG_info);
   ...
}
```

<그림 9> Layout module 구성

```
void XfvManager_setScreenSize
(ScreenSize tResizeScreen)
{
   XFV_CFG.ResizeScreen = tResizeScreen;
}
ScreenSize XfvManager_getScreenSize(void)
{
   return XFV_CFG.ResizeScreen;
}
void XpageViewer_setScaler
         (XpageViewer *viewerPtr,
          XpageViewer_Scaler scaler);
{
   xfv_SCR_SIZE_SAVE.width = pResizeinfor->width;
   xfv_SCR_SIZE_SAVE.height = pResizeinfor->height;
   ...
}
```

<그림 10> Rendering module 구성

4) Display

Display는 하드웨어 플랫폼에 따라서 실제 screen에 layout manager에서 전달 받은 data를 display하는 기능을 제공한다.

그리고, 모바일단말기에서 제공되는 display module에 설정되어 적용된 target의 메모리 DC layout

module에서 넘겨온 data을 적재하여 화면 DC로 display한다. 이를 위하여 display를 구성하며, 메모리 DC에 비트맵에 대한 정보를 생성하고 메모리 DC에 비트맵 정보 저장 영상 정보를 화면 DC로 전송하고 생성한 DC정보를 삭제한다.

5) 메모리, 타이머 관리

효율적인 메모리 관리와 시스템 가독성을 고려하여 file agent에서 페이지별 loading 방식을 취하여 메모리를 관리한다. 그리고, 전체 페이지를 loading하지 않고 우선적으로 첫 페이지를 loading하는 방식으로 이후 background에서 페이지를 추가하는 방식을 취하고 있다.

File loading시 설정된 사이즈를 넘어설 경우 out of memory에 관한 error처리 event를 호출한다. 그리고 적용된 플랫폼의 할당된 메모리에서 FA data buffer, DRL table, code 실행 영역까지 할당하여 memory를 관리한다. 또한 문서 data를 page당 FA에서 해석하여 FA data buffer에 적재한 후 DRL에서 page당 DRL table이 생성된다. 이후 FA data buffer를 clear하여 memory영역을 확보하고, timer를 이용하여 event를 전달한다. 이때 event를 받은 X-File viewer는 사용할 메모리를 확보하고 필요한 timer를 요청한다. 그리고 작업이 끝날 때까지 필요한 timer를 계속 호출한다. 이를 위한 메모리 구성 모듈은 <그림 11>과 같다.

```
void XfvManager_SetMemoryUsage(UINT16 nUsage)
{
   if(nUsage > total_heap_Memory ) return;
   Xfv_CFG.nMemoryUsage = nUsage;
   ... }
```

<그림 11> 메모리 구성 모듈

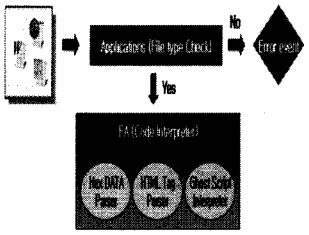
6) 파일 시스템 관리 모듈 구성

파일 시스템 관리 모듈은 H/W platform 및 RTOS에 따라 적용 target 파일 시스템에 종속적으로 구성된다. 그리고, 적용된 플랫폼 target에 지원하는 FFS(Flash File System)을 사용하여 FFS 구성하고, file system의 논리적인 operations과 file directory을 유지관리, garbage collection 및 allocation을 관리하는 기능을 수행한다.

7) X-file viewer 동작

File type의 지원 여부의 판단은 UI 부분에서 결정된다. 이 때 선택된 파일의 확장자가 TEXT, HTML, PDF일 경우 X-File viewer engine을 시작한다. 만약 지원되지 않는 File type일 경우 X-File viewer의 engine을 시작하지 않고 "Not supported file format"로 display하게 되며 error 처리한다.

X-File viewer에서 지원하지 않는 file format의 경우 UI부분에서 xfile Document_error() 함수에 error 메시지를 요청하게 되어 해당되는 error 메시지를 판단 후 상황에 맞는 error를 display 하도록 처리하였다. 지원되지 않는 file type의 처리는 <그림 12, 13>과 같다.



<그림 12> 지원하지 않는 file type 처리

```
void A_xfvDocument_error
(A_xfv_Context *a_xfvContext, XfvDocumentError  error)
{  ...
 switch (error)
  {
    case 0 :
    mmi_MessageBoxString(MB_ICON_WARNING,
    ker_get_name(LABEL_XFV_ERR_UNSUPPORTED),
                  NILSTR);
    break;
    case 1 :
    ...
    default:
    mmi_MessageBoxString(MB_ICON_WARNING,
      ker_get_name(LABEL_XFV_ERR_INTERNAL),
                  NILSTR);
    ...
    break;   } }
```

<그림 13> Error 처리 작업

Applications에서 start event를 X-File viewer engine에 요청 하게 되면 engine에서는 원본 Document를 Loading 하기 전 초기화를 수행한다. <그림 14>는 page count init, document 메모리 확보, 확대/축소 초기화, Flag Init 등의 작업을 수행하는 모

듈이다.

초기화를 마치는 시점에 X-File app start를 통해 파일을 loading한다. File loading (재구성) 후 X-File viewer APP start를 실행하면 FA(File Agent) module이 활성화 되면서 원본 document를 분석하고, 분석된 data를 script화하여 document의 파일 타입에 따라 <그림 15>와 같이 memory에 적재한다.

```
static void Xfv_Init( void )
{  /* Initialize Status Flags */
 XfvManager_SetStatus( XFV_RUNNING | XFV_MEMORYALLOC
                     | XFV_KEYPRESSED | XFV_UPDATED
                     | XFV_WAITUPDATE | XFV_ERR_EXIT
                     | XFVF_HIDDEN | XFV_PIVOT
                     |  XFV_FULLSCREEEN
                     | XFV_SCREENPANNING | XFV_SUBSAMPLED
                     | XFV_DOCLOADED | XFV_SCPRESIZED
                     | XFV_INITCOMPLETED , XFV_FLAG_CLEAR );
  /* Reset all information about the documents */
   XfvManager_SetPageInformation(0,0);
   XfvManager_SetTimerInfo(0, 0);
   Xfv_UIStoreLastestXfvScreenInfo( NULL, 0, 0);
   ...
   Min_Zoom_Reached = FALSE;
   Max_Zoom_Reached = FALSE;     ....}
```

<그림 14> 초기화 작업

```
int XfvApp_loadDocument( void *fileContents,
      size_t fileLength, char *fileExtension )
{
   FileCommand fileCommand;
   static char  fileName[32];
   ...
   /* Parameter validity check */
   if( fileContents == NULL ||
       fileExtension == NULL ||
       fileLength == 0 )
   {
      return 0;
   }
   ...
   /* create RAM filename */
   fileName[0] = 0;
   fileCommand.fileContents      = fileName;
   fileCommand.fileLength        = 0;
   fileCommand.fileExtension[0] = 0;

   return Xfv_loadDocument
   (Context , &fileCommand, NULL);
}
```

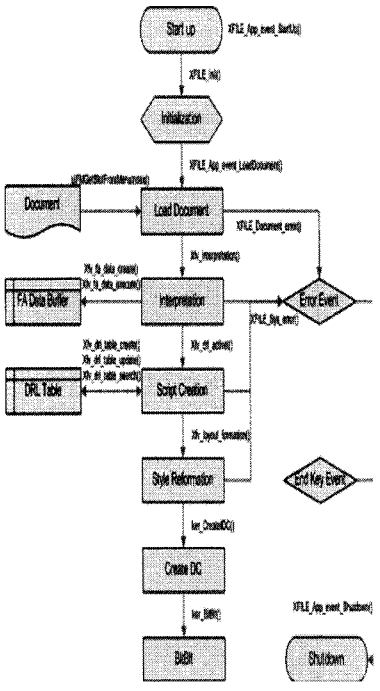
<그림 15> Document Loading 작업

Key event를 통해 screen view영역에서 페이지 이동, 확대/축소, 전체화면 전환과 전체화면, 회전을 할 수 있고, key manager 함수에서 key index값을 획득하여 확대 및 이동 등 전체화면에 대한 동작을 구성하게 된다. Screen view의 동작을 수행하는 모듈은 <그림 16>과 같다.

```
UINT XFV_Key_Management(UINT32 key_index)
{
   .......
   if(key <= XFV_Key_Up && key >= XFV_Key_Right)
   {
      .......
      XFV_event_Key(key, FALSE)
   }
   if(key <= XFV_Key_Zoom_in && key >= XFV_Key_Zoom_out)
   {
      .......
      XFV_event_Key(key, TRUE)
   }
   ........
   return 0;
}
```

<그림 16> 화면의 이동, 페이지이동, 확대/축소, 전체화면 회전 작업

X-file viewer의 수행순서 흐름도는 <그림 17>과 같다.



<그림 17> X-file viewer 수행순서 흐름도

4. 실험 및 분석

X-File viewer의 기능을 테스트하기 위한 환경은 <표 1>과 같다.

<표 1> 개발 및 테스트 환경

(a) 개발 환경

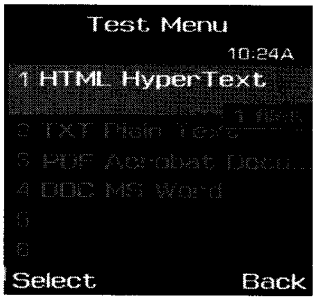
OS	Windows XP
Target	GSM 단말기
Core	ARM9
Utils	Optiflash, Optitrace

(b) 테스트 환경

- X-File Viewer 모듈은 OS 별로 LIB 형태로 제작.
- X-File Viewer 모듈은 Memory, Timer, Display를 위한 외부 인터페이스를 제공.
- 인터페이스를 통하여 외부 OS와 필요한 정보를 제공하고 받음.
- 각각의 LIB는 독립적으로 Task 생성 및 Memory 관리.
- 각 플랫폼상의 UI와 연계되는 Interface Application 개발.
- Application에서 요구하는 파일을 X-File View Module이 가공 후 재 전달
- X-File Viewer Interface Application 과 X-File Viewer Module 라이브러리를 이용하여 샘플파일 테스트

X-File viewer 프로그램은 다음과 같이 수행된다.

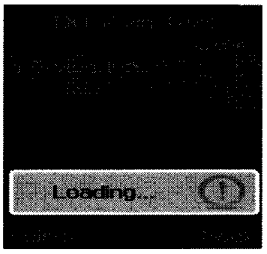
X-File viewer 프로그램 시작 후 X-File viewer start가 되면 문서를 선택할 수 있는 메뉴가 <그림 18>과 같이 구성된다.



<그림 18> X-File Viewer First Page

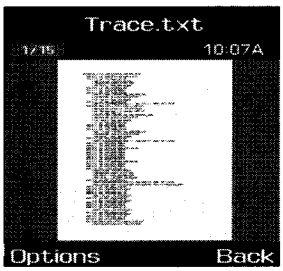
<그림 18>에서 지원하는 문서 format(TEXT, HTML, PDF)을 선택할 수 있다.

<그림 18>에서 지원하는 문서를 선택하였을 경우 X-File viewer 엔진이 start되며 모든 초기화 작업을 마친 후 <그림 19>와 같은 화면을 통해 해당 agent가 start 한다.



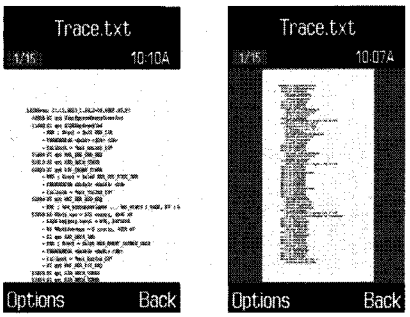
<그림 19> 문서 Loading

<그림 20>의 경우 성공적으로 문서를 open한 사용자가 조작(Next Page, Zoom in-out)을 할 수 있도록 대기 중인 상태이다.



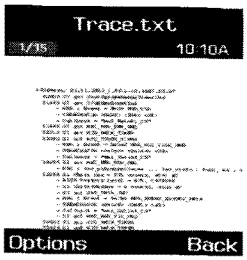
<그림 20> 문서 Loading 완료

<그림 21>은 사용자가 단말기의 key를 이용하여 문서를 확대한 경우 정의된 key event로 open한 문서를 확대/축소 동작을 할 수 있도록 한다.



<그림 21> 확대/축소

<그림 22>는 사용자가 확대 한 상태에서 문서의 아래 부분을 보기위해 정의된 key를 사용하여 문서를 down한 경우이다.

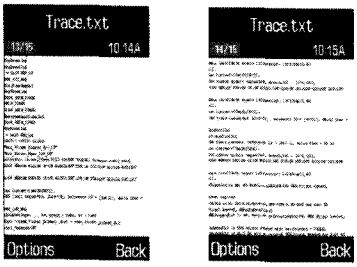


<그림 22> 문서 Down UP

<그림 23>은 정해진 key를 사용하여 문서를 rotate 한 경우로써 현재 출시되는 단말기의 경우 가로 지원이 되는 경우가 많아 rotate 기능을 부가하였다.

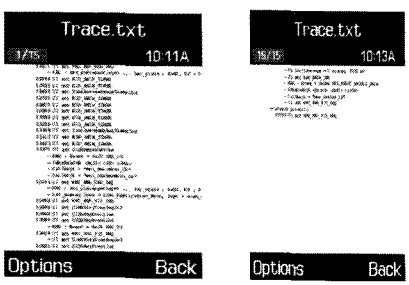
<그림 23> 문서 rotate

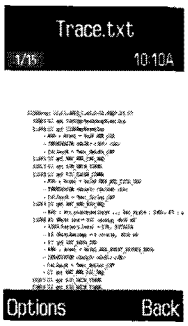
<그림 24>는 사용자가 정의된 key를 이용하여 다음 페이지와 이전 페이지로 이동하는 경우 이다.



<그림 24> Page Next, Before

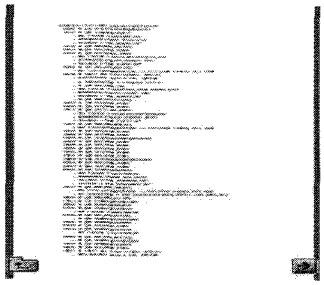
<그림 25>는 사용자가 key를 이용하여 문서의 제일 마지막으로 이동하는 경우이다.





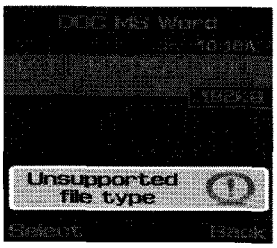
<그림 25> Page 첫 번째와 마지막으로 이동

<그림 26>은 사용자가 "#" key를 이용하거나 menu options의 full view를 선택했을 경우 보이는 이미지이다.



<그림 26> Full view

<그림 27>은 사용자가 단말기의 메뉴에서 지원하지 않는 type의 문서를 open할 경우 display하는 error 메시지이다.



<그림 27> Unsupported file type

5. 결 론

e-Business 유저들의 경우 많은 이동성과 장소에 구애 받지 않고 사용할 수 있는 단말기를 필요로 한다. 본 논문에서 제안한 X-File viewer는 이를 충족시

키기 위해 이동이 편한 작은 단말기(휴대폰, PDA등)에 장착이 가능하고 독립적인 모듈에 동작하는 엔진으로 OS의 연동에 따라서 유연하게 장착이 가능한 엔진이다.

PC에 있는 문서를 외부 인터페이스를 이용하여 단말기로 데이터 전송이 가능하며 단말기 자체에서 외부 기기와 연결하는 기능이 있을 경우 노트북 보다 작은 단말기에 문서를 복사하여 프레젠테이션으로 활용이 가능하다. 작은 단말기의 장점을 이용하여 시간과 장소에 구애 받지 않고 확인이 가능하며 일반 사용자들이 콘텐츠를 다운 받아서 음악을 듣거나 이미지를 viewer 할 수 있는 것처럼 문서를 단말기에 저장하여 필요할 때 마다 활용이 가능하다.

또한 단말기의 통신을 활용하여 MMS등을 이용한 문서 공유가 가능하며 사무실이나 PC가 없는 장소에서도 단말기의 문서를 필요한 시기에 단말기의 통신을 이용하여 서로 공유가 가능하다.

참 고 문 헌

[1] http://www.picsel.com/

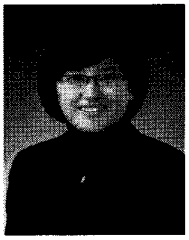
[2] Donato Malerba, Floriana Esposito, Francesca A. Lisi, and Oronzo Altamura. Automated discovery of dependencies between logical components in document image understanding. In ICDAR ''01. pp. 174 - 178.

[3] Nicholas Chen, Francois Guimbretiere, Morgan Dixon, Cassandra Lewis, and Maneesh Agrawala. Navigation techniques for dual-display e-book readers. In CHI '08. pp. 1779 - 1788.

[4] Morgan N. Price, Bill N. Schilit, and Gene Golovchinsky. Xlibris: The active reading machine. In CHI '98. pp. 22 - 23

[5] Shengdong Zhao, Pierre Dragicevic, Mark Chignell, Ravin Balakrishnan, and Patrick Baudisch. Earpod: Eyes-free menu selection using touch input and reactive audio feedback. In CHI '07. pp. 1395 - 1404.

[6] George Buchanan and Tom Owen. Improving navigation interaction in digital documents. In JCDL '08. pp. 389 - 392.



하 경 주 (Kyeoung Ju Ha)

- 정회원
- 경북대학교 컴퓨터공학과 공학사
- 경북대학교 컴퓨터공학과 공학석사
- 경북대학교 컴퓨터공학과 공학박사
- 한국전자통신기술연구원(ETRI) 부호기술연구부 선임연구원
- 대구한의대학교 모바일콘텐츠학부 교수
- 관심분야 : 모바일 소프트웨어, 임베디드 소프트웨어, SNS, 정보보호

논 문 접 수 일 : 2010년 10월 20일
1차수정완료일 : 2010년 11월 05일
2차수정완료일 : 2010년 12월 10일
게 재 확 정 일 : 2010년 12월 15일