SERA Web-Viewer : 사용자 편의성을 향상시킨 웹 브라우저 설계 및 구현*

조영석** · 김재훈** · 장익현***

요 약

우리는 웹 브라우저 사용자의 편의성 향상을 목표로 4가지 편의 기능이 통합된 브라우저인 SERA Web-Viewer를 개발하였다. 편의 기능으로는 VPV(Visited Page Viewer), APV(Aligned Page Viewer), USC(User Specified Capture)등이 추가되었으며 UCC(User Created Content)의 핵심 기술인 FLV(FLash Video file) 변환을 위한 VAC(Video and Audio Converter)를 부가기능으로 통합하였다. 편의 기능들은 사용자들이 웹 브라우저를 사용하며 가장 필요로 하는 기능들로 웹 브라우저의 사용 빈도가 잦고 사용 시간이 긴 고급 사용자들을 대상으로 조사된 바에 따랐다.

위의 네 가지 통합 기능에 대한 구현 상의 알고리즘과 기술을 제안하였고, 인터넷 익스플로러 6.0의 렌더링(rendering) 기술을 이용한 MDI application 기반의 구현 방법을 코드를 통해 설명한다.

구현된 결과를 134명의 컴퓨터학 또는 멀티미디어 공학을 전공하는 학생들을 대상으로 실시한 설문조사의 결과와 비교하여 통합된 편의 기능들이 웹 브라우저 사용자들에게 실질적으로 도움을 줄 수 있다는 것을 보인다.

Sera Web-Viewer : a Convenience-Featured Web Browser

Youngsuk Cho** · Jaehoon Kim** · Ikhyeon Jang***

ABSTRACT

We developed a convenience-featured Web browser which is intended to enhance Web users' convenience. The integrated convenience functions are VPV(Visited Page Viewer), APV(Aligned Page Viewer), USC(User Specified Capture), and VAC(Video and Audio Converter) which is the most important feature of FLV(FLash Video file) in UCC (User Created Contents). The four functions are considered ad the most needed functions to the Web users and we referred to the opinion of frequent and advanced Web users.

We addressed important algorithms and techniques in terms of the implementation of the above four functions. The implementation methods based on the MDI application using rendering technique same as in Internet Explorer 6.0 are shown with codes.

The results of implementation is compared with the survey conducted on 134 Computer Science and Multimedia Engineering major students. All four integrated functions are considered to be useful.

Key words : User-Oriented Convenience Functions, Visited Webpage Viewer, Aligned Webpage Viewer, Specified Area Capture, Video and Audio Converter

---

* 이 논문은 2006년도 동국대학교 연구년 지원에 의하여 이루어졌음.

** 동국대학교 컴퓨터 · 멀티미디어학과

*** 동국대학교 정보통신공학과

1. 서 론

기존의 웹 브라우저는 사용자로 하여금 방문한 사이트의 도메인 주소를 저장하여 특정 사이트를 손쉽고 빠르게 재방문할 수 있게 해주는 '즐겨찾기' 기능과 방문한 사이트들에 대한 기록을 저장하여 재방문할 수 있게 해주는 '기록' 및 '이동' 기능을 갖추고 있다.

웹 문서를 탐색하거나 웹 문서를 통해 발견한 정보를 관리하기 위하여 사용자는 사이트의 목록을 저장 하거나 자주 방문하는 사이트를 따로 관리하는 등의 노력을 해야만 한다. 또한 웹 사용자들의 '즐겨찾기' 기능 의존도가 높다는 사실은 Georgia Tech의 WWW User Surveys (Georgia Tech Research Corporation, 1998a, b)를 통해서도 나타났다. 하지만 사용자가 미처 저장하지 못한 사이트에 대해서는 다시 사이트를 방문하기 어려우며, 이러한 경우 사용자들은 '기록' 기능을 이용하게 된다. 하지만 '기록' 기능의 경우 방문한 사이트들에 대하여 특정한 규칙이 없이 방문한 기록이 나열되어 있어 자신이 방문한 사이트를 찾기 위하여 여러 번의 방문을 통해 사이트를 다시 찾을 수밖에 없다.

본 논문에서는 웹 브라우저의 기본 기능을 모두 수행함과 동시에 '기록' 기능을 보완하여 방문한 사이트의 주소를 tree형태의 구조화된 텍스트 자료 또는 그래픽 자료를 보고 직접 접근할 수 있도록 도와줄 수 있는 VPV(Visited Page Viewer) 기능을 통합하고 현재 열려있는 페이지 모두를 한눈에 볼 수 있게 하는 APV(Aligned Page Viewer) 기능, 사용자가 원하는 부분만 캡쳐하여 저장하는 USC(User Specified Capture) 및 UCC(User Created Content)의 핵심 기술인 FLV(FLash Video file)변환을 용이하게 하는 VAC(Video and Audio Converter)를 부가기능으로 통합한 SERA Web-Viewer의 구현 및 실험에 대하여 기술 한다.

2. 기존 웹 브라우저들의 비교 및 개선 방향

Internet Explorer를 포함하여 사용자들은 다양한 웹 브라우저를 사용한다. 컴퓨터학을 전공하는 학생을 대상으로 한 설문조사 결과 134명의 응답자 중 Explorer를 제외한 웹 브라우저를 사용해 본 경험이 있는 응답자는 총 42명(31.3%)이며, Firefox, Navigator, Safari, Opera 순이었다. 각 브라우저가 지원하는 기능은 <표 1>과 같다.

〈표 1〉 브라우저 기능 비교

브라우저	차별화 된 기능	공통되는 기능
Explorer 6.0, 7.0	• 이동	• 즐겨찾기(북마크) • 기록 • Explorer 6.0을 제외한 브라우저가 탭 브라우징 지원
Firefox	• 열어본 페이지	
Navigator	• Page Info	
Safari Opera	-	

<표 1>에서 '즐겨찾기'의 경우 모든 브라우저가 유사한 기능을 지원하며, '기록' 기능은 조금씩 다르게 나타났다. <표 2>는 브라우저들의 '기록' 기능을 비교한 자료이다.

〈표 2〉 브라우저 기능 비교

브라우저	기록 기능
Opera, Explorer 6.0, 7.0	메인이 되는 페이지로부터 방문 순서와 관계없이 방문 페이지가 단순 나열됨.
Firefox	모든 방문 페이지 저장 안 됨. 메인이 되는 페이지만 표현.
Navigator	대부분의 페이지가 기록 안 됨.
Safari	재방문한 사이트 기록됨(중복기록) 방문한 페이지의 구별 없이 나열됨.

<표 2>에서 각 브라우저마다 '기록' 기능에는 차이가 있지만, 모든 브라우저가 사용자의 방문에

대한 특별한 규칙을 두지 않고 '기록'을 구성하고 있다.

우리가 제안하는 SERA Web-Viewer는 이러한 규칙성 없는 '기록' 기능을 사용자가 방문한 웹 페이지의 순서에 따라 tree 형태로 구현하며, 방문 웹 페이지 정보를 tree로 구현하여 사용자가 한눈에 자신이 방문한 페이지를 확인할 수 있는 VPV 기능을 통합한다.

3. 연구내용 및 관련분야

SERA Web-Viewer는 기존 Internet Explorer의 렌더링(Rendering) 기술을 이용하기 위하여 MSHTML SDK를 이용한 MFC의 CHtmlView 클래스 기반의 MDI application 브라우저를 생성한다. 또한 VPV와 APV 기능을 위하여 가상 DC(Device Context)를 이용한 화면 캡쳐 방법과 FLV 변환을 위하여 GUI Open Source Project FFmpeg을 이용하여 VAC를 구현한다.

제 2장에서는 SERA Web-Viewer 구현을 위해 사용한 MSHTML SDK, FFmpeg, 가상 DC 화면 캡쳐 등을 설명한다.

3.1 MSHTML SDK

MSHTML은 HTML을 구조적으로 설계한 COM (Component Object Model) 인터페이스 모음이다.

MSHTML은 HTML의 순서쌍 구조를 분석 할 수 있게 해주며 기본적으로 tree형의 자료구조로 HTML의 모든 Tag 조합을 표현해 준다.

3.1.1 MSHTML의 IHTMLDocument2

Document 객체는 MSHTML의 모든 객체의 가장 상위에 있는 객체이다. MSHTML이 제공하는 모든 객체(인터페이스)는 일부 서비스 프로바이더를 통해 얻어오는 인터페이스를 제외 하고 Document를 통해 접근이 가능하다. 또한, MSHTML을 자동 QueryInterface를 통해 release 처리를 다루기 위해서는 COM이 필요하다.

〈Code 1〉 COM을 이용한 속성 접근 방법

```
CComQIPtr<IHTMLDocument2> pDoc ;
CComPtr<IHTMLElement> pBody ;
pDoc->get_body(&pBody) ;
CComQIPtr<IHTMLBodyElement>
        pBodyElement(pBody) ;
CComBSTR strBackground ;
pBodyElement->get_background(&strBackground) ;
```

<Code 1>은 COM의 QueryInterface를 이용한 IHTMLDocument2를 통해 HTML의 Body Tag 내의 Background 속성에 접근하는 방법이다. HTML의 모든 Tag 내의 속성은 위와 같은 방법으로 접근이 가능하다.

3.1.2 MSHTML의 IHTMLElement

HTML 파일이 MSHTML을 통해 파싱된 후 내부적인 자료구조에 저장된 후에는 이 개체들을 Element라고 부른다. 즉, HTML 파일에 저장할 때는 Tag라 하고, MSHTML을 통해 파싱된 후에는 Element라 한다.

모든 Tag는 Element로 표현되기 때문에 각 Tag는 IHTMLElement 인터페이스를 지원한다. <Code 2>는 IHTMLElement를 통하여 HTML의 FONT Tag의 Element 변환을 보여준다.

〈Code 2〉 Font Tag의 Element 변환

```
                Font Element 변환
CComPtr<IHTMLElement>pFont ;
CComQIPtr<IHTMLFontElement> pFontElement(pFont) ;
CComBSTR face ;
pFontElement->get_face(&face) ;
```

MSHTML에서 Element를 표현하는 방법은 일단 모든 Element가 공통적으로 제공하는 메소드들은 IHTMLElement로 구현하고, 특정 Tag에만 해당되는 속성일 경우, 해당 인터페이스를 QueryInterface하여 포인터를 넘겨받은 후 처리한다.

3.2 UCC와 FLV 변환을 위한 FFmpeg

2006년 12월 미국의 시사주간지 <타임>이 '2006 올해의 인물'로 '유(You)'를 선정하고 '블로그'나 미디어 영역에서 영향력을 키워가는 평범한 당신이 바로 '올해의 주인공'이라고 발표, 새로운 문화 트렌드로서 UCC(User Created Contents)의 힘을 전 세계에 확인시켰다.

웹 2.0 시대의 핵심기술 중 하나는 바로 UCC의 FLV(FLash Video file)이다. FLV 형태로 된 UCC들은 원본 파일에 접근하기가 어렵고, 공식적인 방법을 통해서만 재배포가 가능하다. 이는 저작권 문제로부터 상당히 자유롭다는 뜻이 되지만, 사용자가 FLV 동영상을 따로 소장하고자 하면 지원해주는 툴이 없어 어려움이 따른다. 또한, Windows Mac 등의 다양한 OS에서 Flash가 지원되는 플랫폼이라면 FLV의 재생이 가능하다. Flash 플레이어가 거의 모든 플랫폼을 지원하기 때문에, 콘텐츠의 공유란 측면에서 매우 큰 장점을 가지고 있다. 또한 압축률이 높으므로 상대적으로 적은 저장 공간으로 높은 화질을 유지 할 수 있다는 장점이 있다.

VAC(Video and Audio Converter)는 사용자가 보다 작은 용량의 FLV 동영상을 소장할 수 있게 하기 위하여 SERA Web-Viewer에 통합된다.

FLV 포맷 변경을 위해 GUI Open Source Project인 FFmpeg을 이용하였으며, 이를 Microsoft의 Visual Studio .net 2003에서 사용하기 위해서는 윈도우즈에 가상 Unix 환경을 구성하여 DLL 파일을 생성해야 한다. 이러한 환경 구성에 관해서는[1]의 FFmpeg on Windows 문서에 상세히 기술되어 있다.

3.3 IHTMLDocument2를 이용한 가상 DC 캡쳐

화면에 Display된 정보를 캡쳐하기 위해서는 우선 ScreenDC를 생성한 후 ScreenDC와 호환되는 메모리 DC를 생성한다. 생성된 메모리 DC에 저장하고자 하는 영역의 이미지를 CBitmap으로 변환하여 생성된 DDB(Device-Dependent Bitmap)를 DIB(Device-Independent Bitmap)로 변환하여 파일에 저장한다.

그러나 이러한 캡쳐 방식은 화면에 display된 정보가 아닌 비활성화 된 top window와 메모리상에는 display가 완료되었으나 모니터상의 display 정보와 일치하지 않는 경우 잘못된 정보를 캡쳐 한다. 따라서 SERA Web-Viewer는 화면의 display 정보가 아닌 IHTMLDocument2를 이용한 가상 DC 화면 캡쳐 방법을 통해 비활성화 된 child frame과, load가 완료 되지 않거나 여러 화면이 중복되어 있는 화면에 대해서도 완전한 웹 페이지(Web Page) 형태로 이미지를 저장하는 방법을 택하였다.

〈Code 3〉 가상 DC를 이용한 화면 캡쳐 방법

```
1. 사용자가 지정한 이미지 size만큼의 rect 생성
2. 가상 Window 창을 생성하여 rect만큼의 size
   생성
3. 접근하고자 하는 웹 페이지의 HTML 소스 코드
   획득
   CComPtr<IWebBrowser2> m_pBrowser ;
   // URL에대한 HTML 소스코드 획득
   IDispatch *pDoc = (IDispatch *) NULL ;
   HRESULT hr = m_pBrowser
    ->get_Document(&pDoc) ;
   m_pBrowser->get_Document(&pDoc) ;
4. HTML 전체 소스 코드 중 <body> 부분만을
   추출
   IHTMLElement *pElement
           = (IHTMLElement *) NULL ;
   pDoc->get_body(&pElement) ;
5. body부분에 대한 QueryInterface를 통해 정보
   저장
   IHTMLElementRender *pRender
   = (IHTMLElementRender *) NULL ;
```

```
    pElement->QueryInterface
    (IID_IHTMLElementRender, (void **)
          &pRender);
6. 가상 DC에 DrawToDC() 메서드를 통해
    CBitmapDC에 Render 작업
    // srcSize는 처음 화면 캡쳐를 위해 지정한 size
    CBitmapDC destDC(srcSize.cx, srcSize.cy);
    pRender->DrawToDC(destDC);
7. CBitmapDC의 컨트롤을 완료 후 CBimap에
    메모리 내용을 저장(캡쳐 화면 정보)
    CBitmap *pBM = destDC.Close();
8. GDI PLUS의 FromHBITMAP() 메서드를 통해
    파일저장 위한 Bitmap으로 변경
    Bitmap *gdiBMP = Bitmap :: FromHBITMAP
    (HBITMAP(pBM -> GetSafeHandle()), NULL);
9. 이미지 사이즈 지정
    // outputSize는 사용자가 지정한 save하기
       위한 size Image *gdiThumb = gdiBMP ->
    GetThumbnailImage(outputSize.cx,
                      utputSize.cy);
10. 메모리 내용을 파일로 저장
    CLSID m_encoderClsid;
    gdiThumb -> Save((WCHAR *)(LPCTSTR)
    fsDest.GetFullSpec(), &m_encoderClsid);
```

<Code 3>은 IHTMLDocument2를 이용한 가상 DC 캡쳐 방법이다.

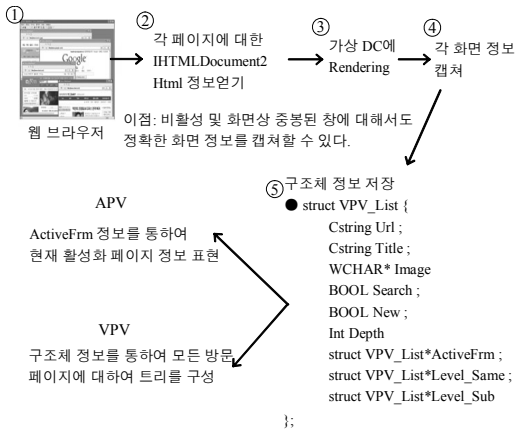
4. SERA Web-Viewer의 구현

SERA Web-Viewer는 웹 브라우저의 사용자 편의성과 효율을 고려하여 VPV(Visited Page Viewer), APV(Aligned Page Viewer), VAC(Video and Audio Converter), USC(User Specified Capture)등 4가지 기능을 통합한다. VAC는 UCC의 FLV 변환을 대비하고 사용자에게 저용량·고화질의 파일 포맷을 제공하며, 각종 동영상 및 오디오 파일을 다른 형태의 포맷으로 변환해준다.

VPV, APV, USC는 사용자들의 웹 서핑에 편의성 제공을 위해 제작된 기능들이다. 특히 VPV는 기존 웹 브라우저에서 사용하던 '기록' 기능을 보다 효과적으로 발전시킨 기능으로 제 4장에서는 각 기능의 실제 구현 알고리즘과 source code를 통해 작성 방법을 제시한다.

4.1 VPV(Visited Page Viewer)

VPV 기능은 MDI application 구조의 웹 브라우저 뿐만 아니라 SDI application 구조의 웹 브라우저에도 적용될 수 있다. VPV의 기본 개념은 익스플로러의 '기록' 기능을 보완 하여 사용자에게 자신이 방문한 모든 웹 페이지 중 다시 찾고자 하는 웹 페이지를 한번에 찾고자 하는 것이다.



(그림 1) APV, VPV를 위한 흐름도

(그림 1)은 VPV와 4.2에서 설명할 APV의 기본 구조를 저장하기 위한 흐름도이다. 시스템 자원을 효율적으로 이용하기 위하여, 웹 페이지 로딩이 완료되는 시점 ①을 확인한다. 웹 페이지의 로딩이 완료되면 ②에서 웹 페이지의 IHTMLDocument2 정보를 얻는다. 이 정보를 통해 ③에서 가상 DC에 웹 페이지의 렌더링이이루어지고, ④에서 화면 캡쳐가 이루어진다. ④의 캡쳐 정보와 웹 페이지의 url, title 등의 정보가 ⑤의 구조체에 저장된다.

MDI application 구조에서 child frame의 id를 얻을 수 있기 때문에 생성된 child frame의 id를 얻고 이에 따른 노드의 head 정보에서 active frame

링크 노드의 정보를 통해 APV 기능 구현이 가능하다. 또한 VPV는 여기서 저장된 모든 정보를 통해 해당 frame의 모든 방문 페이지에 대하여 tree를 구성한다. 이때 그 경로가 main인 웹 페이지를 방문 시 그 웹 페이지를 또 다른 root 시작점으로 생성하고, 웹 페이지 내의 또 다른 웹 문서 페이지를 방문 시 root의 하위 노드로 표현한다. root가 되는 url 변경 시점은 다음의 경우가 있다.

(ⅰ) 사용자가 URL의 직접적으로 변경하여 이동
(ⅱ) 포털 사이트의 검색 결과에 의한 사이트 이동
(ⅲ) 즐겨찾기를 통한 저장된 주소를 지정하여 사이트를 이동한 경우

메인이 될 수 있는 url 주소로 변경되는 시점인 경우(ⅱ)와 경우(ⅲ)에 대해 SERA Web-Viewer에서는 새로운 frame이 생성되어 렌더링이 이루어진다.

VPV는 각각의 frame을 새로운 tree로 구성한다. 즉 각 frame에 대하여 이동한 경로의 정보가 따로 저장된다. 따라서 이유 (ⅰ)만이 main이 될 수 있는 url 주소의 변경 시점이 될 수 있다.

〈알고리즘 1〉 VPV 알고리즘

```
if ( 웹 페이지 로딩 완료 ){
  SearchLocation() ;    // 방문 기록 확인
  if ( 방문기록이 없다 ){
    CreateImage() ; // 가상DC 통한 이미지 파일 생성
    // 처음 프로그램 구동시 로딩된 웹 페이지의 경우
    if (최초 방문){
       Head -> ActiveFrm = Head -> Level_Sub ; }
    else { // 웹 페이지를 2번 이상 방문한 경우
      if (기존 노드에서 링크가 생성되는 경우) {
      if (기존 노드의 하위 노드가 없는 경우) {
        Head -> ActiveFrm
        = Head -> ActiveFrm -> Level_Sub ; }
      else if (기존 노드의 하위 노드가 있는 경우) {
        SearchInputLocation
        (Head -> ActiveFrm -> Level_Sub) ; } }
    else if (기존 노드가 아닌 새로운 루트 생성 경우)
           {
     SearchInputLocation(Head->LevelLow) ; } }
  Head -> ActiveFrm = 현 Link 정보} }
```

메인 url의 이유 (ⅰ)에 근거 한 VPV 구현 알고리즘은 <알고리즘 1>과 같다. <알고리즘 1>은 VPV와 APV 기능을 위해 필요한 정보 즉, 자신이 방문한 사이트에 대한 웹 페이지 정보를 수집한다.

각 노드는 웹 페이지의 Url과 Title 정보, 캡쳐한 이미지의 파일 이름, 해당 노드의 깊이 및 노드에서 연결되는 하위 레벨과 동일 레벨 노드의 정보이다. New 변수를 통해 각 노드가 ActiveFrm 노드에서 연결되어야 하는지 새로운 root를 생성하는지 판단하게 되며, Search 변수를 통해 이미 방문한 기록이 있는지 여부를 알 수 있다.

<알고리즘 1>의 SearchLocation() 메소드는 구성된 linked list 내의 노드 정보 중 방문 기록을 확인하는 메소드로서 <알고리즘 2>에 상세히 설명했으며, SearchInputLocation() 메소드는 구성된 linked list 내의 노드 정보 중 새로운 노드를 생성하여 삽입해야 할 위치를 판단하는 메소드로서 <알고리즘 3>과 같다.

<알고리즘 2>와 <알고리즘 3>은 기본적으로 입력받은 파라메터 노드를 재귀적으로 구성한다. <알고리즘 2>는 Head를 파라메터로 시작하여 모든 방문 기록 중 현재 페이지의 url과 비교하여 탐색한다. 만약 방문 기록이 없을 경우 새로운 노드의 삽입 위치를 결정하기 위하여 <알고리즘 3>이 적용된다. <알고리즘 3>은 현재 자신이 삽입될 부모노드를 파라메터로 시작하게 되며, 모든 동일 레벨의 마지막 링크에 삽입 위치를 결정해 주게 된다.

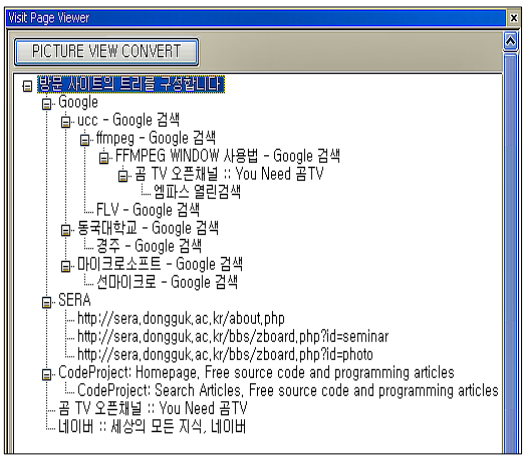
〈알고리즘 2〉 SearchLoction 알고리즘

```
// 링크가 현재 페이지와 일치할 경우
if ( 현재 Url과 Link의 Url이 같다 ) {
     Head->ActiveFrm = Link ; }
else { // 링크가 현재 페이지와 같지 않을 경우
       // 깊이 우선 탐색 방법을 이용한 재귀
       if ( Link의 하위 레벨이 있다 ){
         SearchLocation(Link -> Level_Sub) ; }
       else if ( Link의 하위 레벨이 없다 ) {
         SearchLocation(Link ->
Level_Same) ; }}
```

〈알고리즘 3〉 SearchInputLocation 알고리즘

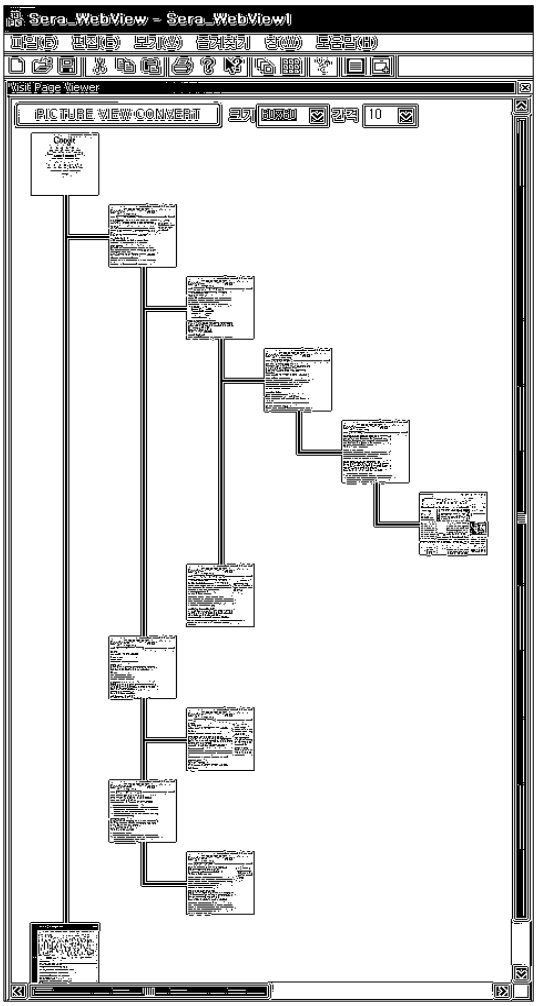
```
// 넓이 우선 탐색 방법의 부분 적용을 이용한 재귀
if ( Link와 같은 레벨의 노드가 없다 ){
    Head -> ActiveFrm = Link - >Level_Same ; }
else ( Link와 같은 레벨의 노드가 있다 ) {
    SearchInputLocation(Link -> Level_Same) ; }
```

<알고리즘 1>에서는 웹 페이지의 로딩이 완료되는 시점을 판단하는 것이 중요하다. SERA Web-Viewer는 익스플로러의 렌더링 기술을 활용하기 위해 MFC의 CHtmlView Class를 상속받고, CHtmlView의 OnNavigateComplete2() 메소드는 이러한 웹 페이지의 로딩 완료 시점을 판단해 준다.



(그림 2) VPV의 Title을 이용한 Tree View

(그림 2)는 생성된 노드들을 TreeControl을 통해 각 웹 페이지의 title을 보여주는 화면이다. root가 되는 사이트는 Google, SERA, CodeProject, Gom TV, Naver의 5개 사이트이며 각 사이트에서 방문한 순서대로 tree가 구성되었다. 각 사이트의 Title 정보를 통해 사용자는 자신이 방문한 사이트들을 한눈에 볼 수 있고, 또한 tree view에서 각 사이트 클릭을 통해 한 번에 해당 사이트로의 이동이 가능하다. title 정보는 방문 사이트의 제목이나 url이기 때문에 사이트제목이 애매모호 하거나 hyper-link된 같은 웹 페이지 내에서 같은 제목을 사용하는 경우, 또는 방문했던 웹 페이지의 url을 기억하기 어려울 때에는 tree view의 각 page를 축소한 그림으로 확인할 수 있게 한다. (그림 3)의 각 노드는 방문했던 page를 캡쳐하여 축소한 것으로 확대·축소가 가능하여 기억을 보다 쉽게 되돌릴 수 있게 하였다. (그림 2)의 tree view 상단에 'Picture View Convert' 버튼을 클릭하여 사용자는 해당사이트의 캡쳐 된 화면을 tree로 볼 수 있다.



(그림 3) VPV의 Imagae Tree View

(그림 3)은 이러한 VPV 기능의 image tree view 이며 사용자의 편의를 고려하여 3가지 크기의 이미지를 제공한다. (그림 3)의 상단에 있는 '크기'는 캡쳐된 화면의 크기이며, '간격'은 이미지와 이미지 사이의 pixel 값이다.

VPV는 40x40, 60x60, 80x80의 이미지 크기가 제공되며 (그림 3)은 80x80으로 나타낸 화면이다. 한 화면에 나타나지 않는 정보는 우측과 하단의 스크롤바를 통하여 볼 수 있으며, 이미지의 크기는 축소 및 확대가 가능하다.

방문 기록 재탐색 시간은 사용자가 방문한 웹 페이지가 n개일 경우, 이미 방문했던 페이지를 재방문시, 재탐색 되어야 할 웹 페이지의 수는 O(n)이다. 그러나 VPV 기능을 사용할 경우 tree에서 방문했던 웹 페이지를 찾아 클릭함으로써 탐색 횟수는 O(1)이 되어 사용자의 탐색 시간이 현저히 줄어듦과 동시에 인터넷의 부하를 줄일 수 있다.

4.2 APV(Aligned Page Viewer)

APV 기능은 MDI Application 기반의 브라우저를 이용하여 다수의 브라우저를 활성화하여 사용시 각각의 브라우저에 open된 웹 페이지에 대한 시각적 정보를 제공하는 기능이다.

〈알고리즘 4〉 APV 알고리즘

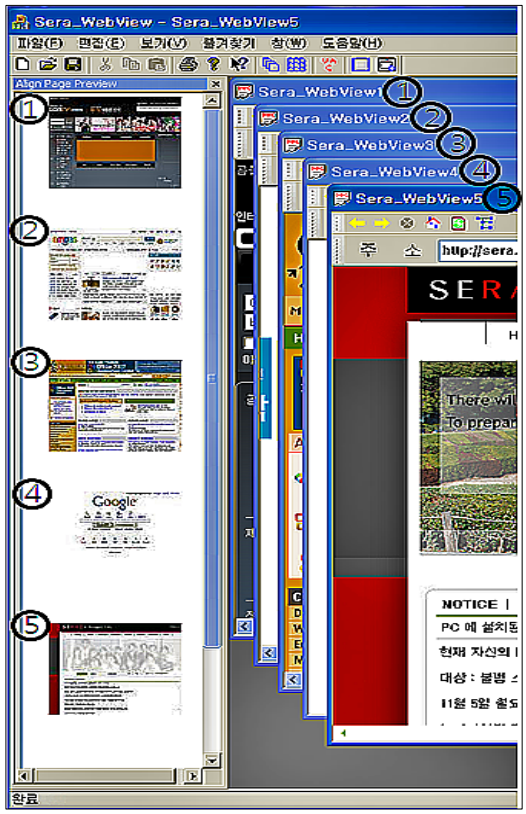
```
// 메인 프레임 포인터 획득
Get MainFrame Pointer
// child frame이 존재하지 않을 때 까지
while (!(MainFrame -> ChildFrame Id)) {
    // child frame id 획득의 경우
    if (ChildFrame Id) {
        // child frame의 ActiveFrm 링크 이미지
           로드
        LoadImage
        (ChildFrame Id -> HEAD -> ActiveFrm
        -> Image)
        DrawImage() ; }
    GetNextChildFrame() ;
        // 다음 child frame의 id 획득 }
```

APV 기능은 4.1의 (그림 1) ⑤에서 active frame 정보를 통해 구현된다고 설명한 바 있다. <알고리즘 4>는 APV 기능을 화면에 보여주기 위한 알고리즘이며 APV 알고리즘이 정상적으로 적용하기 위해서는 다음의 두 가지 조건이 필요하다.

(ⅰ) 각 child frame은 VPV_List 구조체에 대한 HEAD를 가져야 한다.

(ⅱ) 각 child frame은 자신의 HEAD를 통해 방문한 모든 웹 페이지의 링크를 가지고 있고 마지막으로 방문한 웹 페이지가 ActiveFrame 포인터로 링크 되어 있어야 한다.

VPV_List 구조체 및 조건(ⅰ)과 조건(ⅱ)를 위한 알고리즘은 4.1의 VPV 알고리즘에 자세히 기술되어 있다.



(그림 4) APV의 cascade 정렬

(그림 4)는 APV 실제 화면의 일부이다. (그림 4)는 웹 페이지들이 cascade 형태로 정렬되었고 title 형태의 정렬도 가능하다. 각 child frame은 겹쳐있거나 작아져 있지만 웹 페이지 전체 화면이 캡쳐되어 좌측 form에 출력 된다. 여기서 사용된 방법이 3.3에서 소개한 IHTMLDocument2를 이용한 가상 DC 캡쳐 방법이다. 또한 (그림 4)는 각 child frame의 id 순으로 위에서 아래로 즉 그림의 번호 순으로 출력 된다.

SERA Web-Viewer를 사용하여 웹 surfing 중 현재 사용 중인 frame을 제외한 나머지 frame들은 비활성화 상태이거나 사용 중인 최상위 active frame에 의해 다른 frame들이 가려져 있는 상태이다. APV 기능은 어떠한 frame에 어떠한 사이트가 로드 되어 있는지 한눈에 확인 할 수 있게 한다.

4.3 VAC (Video and Audio Converter)

VAC는 FLV 포맷 변환 툴로 FFmpeg을 Windows용으로 컴파일 한 DLL을 통하여 Visual Studio .net 2003에서 제작되었다. VAC에서 지원되는 Video Codec과 Audio Codec은 <표 3>과 같다.

〈표 3〉 VAC에서 지원하는 File Format

Video Cod	AVI, FLV, MOV, MPEG, SWF
Audio Codec	MMF, MP3, OGG, WAV, WMA

VAC와 다른 포맷 변환 프로그램의 차이점은 다음과 같다.

(i) FLV video format 지원

(ii) video와 audio를 통합한 converter

VAC의 최대 장점은 FLV를 지원한다는 것이다. FLV는 대부분의 플랫폼을 지원하며, 압축률이 높아 상대적으로 낮은 저장 공간으로 높은 화질을 유지 할 수 있다는 장점이 있다. <표 4>는 avi 및 mpeg format을 flv로 convert한 결과이다.

〈표 4〉 FLV 변환 결과

원본 정보	FLV 정보
test.avi(469.575KB) size : 624 * 336 bitrate : 844kb/s frame rate : 23.98fps	test.flv(346,911KB) size : 640 * 480 bitrate : 800kb/s framte rate : 24fps
test2.mpg(381,173KB) size : 720 * 480 bitrate : 6499kb/s frame rate : 29.97fps	test2.flv(332.914KB) size : 800 * 600 bitrate : 6400kb/s frame rate : 30fps

<표 4>에서 원본과 유사한 video option을 통해 FLV로 변환한 결과 높은 압축률을 보임을 알 수 있다.

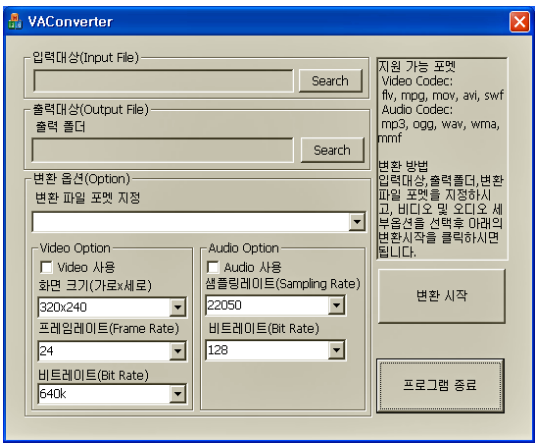
VAC는 Video option에서 size와 frame rate, bit rate를 조절할 수 있으며, Audio option에서 sampling rate와 bit rate의 조절이 가능하다.

다음은 Visual Studio에 avutil.dll avcodec.dll avformat.dll을 추가한 후 필요로 한 메소드를 사용한 방법이다.

```
extern “C” __declspec(dllimport)
    AVCodec *avcodec_find_encoder(enum
    CodecID id) ;
extern “C” __declspec(dllimport)
    AVCodec *avcodec_find_decoder(enum
    CodecID id) ;
extern “C” __declspec(dllimport) int
    avcodec_encode_video(struct
    AVCodecContext *avctx, uint8_t *buf, int
    buf_size, const struct AVFrame *pict) ;
extern “C” __declspec(dllimport) int
    avcodec_decode_video(AVCodecContext
    *avctx, AVFrame *picture, int
    *got_picture_ptr, uint8_t *buf, int
    buf_size) ;
    ...
```

위의 메소드들은 Video Codec에대한 encoder 및 decoder를 포함 한다.

(그림 5)는 VAC의 UI이며 SERA Web-Viewer의 툴바 아이콘을 통해 실행된다.



(그림 5) VAConverter의 UI

4.4 USC(User Specified Capture)

USC는 Web page에서 사용자가 원하는 부분만을 캡쳐하여 저장하는 기능이다. 타 웹브라우저 사용 시 자신이 원하는 부분을 캡쳐하기 위해서는 Print Screen 키를 이용하여 전체 화면을 캡쳐한 후 그림편집 툴을 이용하여 편집해야 하는 불편함이 있다. USC는 사용자가 툴바 아이콘을 클릭하여 실행 시킨 후 캡쳐 하고자 하는 부분을 마우스를 드래그하여 캡쳐할 수 있도록 SnaperHelper 라이브러리가 제공하는 GetRegionImage() 메소드를 통해 영역을 지정하여 JPG와 BMP 2가지 format으로 저장할 수 있다. 이 툴은 기본적으로 제공하는

〈Code 4〉 User Specified Capture 방법

```
1. GetRegionImage() 메소드를 통해 사용자가
   캡쳐하고자 하는 영역을 얻어온다.
      HBITMAP *hImage = new HBITMAP ;
      *hImage = GetRegionImage(&Size) ;
2. GetFullPathFileName() 메소드를 통해
   저장하고자 하는 경로와 파일명을 얻어온다.
      CString sFileName ;
      GetFullPathFileName(this, sFileName)
3. ImageCapture 클래스의 Capture() 메소드를
   통해 파일로 저장한다.
      CaptureScreen ImageCapture ;
      // ImageCapture 객체
      ImageCapture.Capture(*hImage,sFileName) ;
```

SERA Web-Viewer내의 frame에 display된 웹 페이지 뿐만 아니라 SERA Web-Viewer가 구동된 Client의 디스플레이 장치 즉, 모니터 화면 내의 어떠한 내용도 캡쳐가 가능하다. <Code 4>는 이러한 USC를 구현하기 위한 방법이다.

<Code 4>에서 제시된 방법의 주요 클래스는 ImageCapture 클래스이다. ImageCapture 클래스는 디스플레이 장치의 DDB 정보를 DIB로 변환하여 파일에 저장하는 캡쳐의 전반적인 작업을 담당하는 클래스이다.

ImageCapture 클래스는 Capture(), DDBToDIB(), WriteDIB()의 총 3개의 메소드로 구성되며 Capture() 메소드는 캡쳐 전체의 과정을 관리, DDBToDIB() 메소드는 Device에 종속적인 Bitmap을 Device에 독립적인 Bimap으로 변경하며, WriteDIB() 메소드는 DDBToDIB() 메소드로부터 생성된 DIB를 파일로 저장한다.

(그림 6) USC 캡쳐영역 지정 화면

(그림 6)은 SERA Web-Viewer에서 웹 페이지를 캡쳐하는 화면이다. USC는 사용자가 지정한 화면이 정확하게 지정되었는지를 확인하고 file로 저장하기 위하여 지정한 영역을 form을 통해 미리 보여준다. 미리보기 기능을 통해 사용자는 지정된 캡쳐 영역이 의도한 대로 인지 확인 할 수 있다.

USC는 '알 툴바'에서도 지원하는 기능이나 웹

브라우저 자체에 내장된 기능이 아닌 툴바의 추가 설치를 통해 웹 브라우저에 docking하여 사용하여야 하므로 불편하다. 우리는 사용자가 브라우저를 따로 취향에 맞춰 구성하지 않고도 사용할 수 있도록 USC를 통합하였다.

6. 결론 및 향후 연구 방향

우리는 웹 브라우저의 '기록' 기능을 보완한 VPV와 사용자 편의성을 고려한 APV, USC, VAC 등을 설문 응답자들의 의견을 수렴하고 반영하여 MS-HTML SDK, FFmpeg, 가상 DC 캡쳐 방법 등을 통해 구현하였다.

134명의 컴퓨터학 또는 멀티미디어 공학을 전공하는 학생들을 대상으로 실시한 설문조사 결과 매일 웹 브라우저를 사용하는 학생이 114명(85.1%)이었으며 이 중 매일 1시간 이상 사용자는 131명(97.8%)이었다. 응답자 가운데 '기록'을 사용해 본 70명(52.2%) 중 55명(78.6%)이 '기록' 기능이 불편하다고 응답하였으며, 이중 42명(76.4%)이 자신이 찾고자 하는 방문 기록을 찾기 어렵다고 응답하였고, 방문 순서가 체계적으로 정리되어 있지 않아 사용하기 힘들다는 의견도 있었다. '기록' 기능 개선을 위해 text tree 또는 image tree가 지원되었으면 하는 응답자가 121명(90.3%)으로 나타나 SERA Web-Viewer의 '기록' 기능을 강화한 VPV가 설문조사 결과에 부응함을 보아 사용자가 보다 효과적으로 웹 브라우저를 사용할 것이라 기대된다.

SERA Web-Viewer를 구현하는 과정에서 가상 DC를 통한 캡쳐 방식에는 문제가 있다는 것이 발견되었다. 이는 웹 표준을 지킨 모든 html 문서에 대해 정확한 캡쳐가 가능하나 flash 등이 html 문서 안에 포함되어 있는 경우에는 로드가 완료되지 않은 시점에서 캡쳐가 이루어져 flash 부분이 표현되지 않는 경우가 발생한다. 이는 display device, 즉 모니터가 아닌 가상 DC를 사용 하여 캡쳐함으로 발생되는 문제이다. 이 문제가 해결된다면 자신이 방문한 웹 페이지에 대해 보다 정확한 정보를 얻을 수 있으므로, VPV 기능은 효율성은 더욱 증가할 것이다.

또한 VPV가 제공하는 tree에 사용자가 의미 있다고 판단되는 page에 대해 check 또는 관리 할 수 있는 기능을 추가하여 사용자가 더 빨리 웹 페이지를 재 방문 할 수 있도록 보완하고 있다.

참 고 문 헌

[1] http://arrozcru.no-ip.org/ffmpeg_wiki/tiki-index.php, FFmpeg on Windows.

[2] 이해영, 인터넷 이용현황 및 이용자 연구 : 한국・미국・영국의 조사 결과 비교 분석, 산업기술정보원, 1999.

[3] P. Berthon, L. Pitt, and R. T. Watson, Re-surfing World Wide Web : research perspectives on marketing communication and buyer behaviour on the world wide web. International Journal of Advertising, Vol. 15, No. 4, pp. 287-301, 1996.

[4] Georgia Tech Research Corporation, GVU's ninth WWW user survey. Graphics, Visualization, and Usability Center, College of Computing, Georgia Institute of Technology, April, Atlanta, GA, 1998a.

[5] Georgia Tech Research Corporation, GVU's tenth WWW user survey, Graphics, Visualization, and Usability Center, College of Computing, Georgia Institute of Technology, October, Atlanta, GA, 1998b.

[6] http://ffmpeg.mplayerhq.hu/ffmpeg-doc.html, FFmpeg Documentation.

[7] ROBERTS, Programming Microsoft Internet Explorer 5, Microsoft Press, 1999.

[8] 문용은, 웹2.0과 UCC. Journal of Economics & Management, 2007.

[9] 홍종필, 인터넷 사용자의 웹 브라우징 행동에서 '즐겨찾기' 이용에 관한 실증적 연구. Journal of advertising, 2004.

조 영 석

1978년 서강대 철학과(문학사)
1988년 Louisiana State University (정보학 석사)
1994년 Louisiana State University (컴퓨터학 박사)
1980년～1984년 한국 후지쯔(주), Systems Analyst
1989년～1994년 Louisiana State University, Computer Analyst
1999년～현재 동국대 컴퓨터·멀티미디어학과 교수
관심분야 : 소프트웨어 재사용, 소프트웨어개발 방법론, 설계패턴, 객체지향 방법론, 정보검색

김 재 훈

2001년～현재 동국대학교 컴퓨터학과 학생
관심분야 : 컴퓨터 네트워크, 멀티미디어 처리

장 익 현

1984년 서울대 계산통계학과 (이학사)
1986년 한국과학기술원 전산학과(공학석사)
1998년 한국과학기술원 전산학과(공학박사)
1986년～1999년 (주)데이콤 책임연구원
1999년～현재 동국대 정보통신공학과 교수
관심분야 : 컴퓨터통신, 분산시스템, 인터넷 응용