제23회 한국정보처리학회 춘계학술발표대회 논문집 제12권 제1호 (2005. 5)

스크린리더 개발 생산성 향상을 위한 개방형 API 설계

이승수, 김석일
충북대학교 전자계산학과
e-mail : ryang@chol.com

Open API Design for Screen Reader

Seungsoo Lee, Sukil Kim
Dept. of Computer Science, Chungbuk National University

요 약

스크린리더는 그 특성상 모든 응용 프로그램을 지원해야 함에도, 타 응용 프로그램의 빈번한 업그레이드 와 스크린리더 개발 인력 부족 등의 이유로 시각장애인들이 자주 쓰는 소수의 응용 프로그램만을 지원하는 것이 현실이다.

본 논문에서는 스크린리더의 개발 생산성을 높일 수 있는 방안으로 개방형 API 를 제안한다. 스크린리더에 개방형 API 를 채택하고 이를 공개함으로써 스크린리더 개발 업체는 주엔진 모듈의 성능 향상에 모든 인력을 투입하여 성능 좋은 스크린리더을 개발할 수 있을 것이며, 프로그램 개발이 가능한 사용자나 자원봉사자들이 개방형 API 에 맞도록 응용 프로그램 지원 모듈을 플러그인 형태로 개발할 수 있을 것이다. 이를 통해 국내에서도 다양한 응용 프로그램을 지원하는 우수한 스크린리더를 개발할 수 있을 것이다.

1. 서론

스크린리더(Screen-Reader)는 윈도우의 메뉴나 리스트, 현재 포커스된 메뉴나 리스트 아이템, 버튼 등의 정보를 읽어주고, 현재 화면의 정보나 이벤트 정보를 읽어주어 시각장애인들이 청각을 이용하여 컴퓨터를 사용할 수 있도록 도와주는 프로그램이다. 따라서 시각장애인의 경우 컴퓨터를 활용함에 있어 스크린리더의 도움은 필수적이라 할 수 있다[2].

현재 미국에서는 Window-Eyes[4]와 Jaws for windows[5]등이 개발되어 있다. 이들 스크린리더는 뛰어난 읽기 기능을 가지고 있어, 시각장애인들이 이를 이용하여 응용 프로그램까지 개발할 수 있다.

국내의 경우 1998 년 이래 소리눈 98, 아이즈 2000, 소리눈 2000, 드림보이스, 이브 포 윈도우, 센스리더 등 수 종의 윈도우용 스크린리더를 개발해 오고 있다. 하지만 아직 안정성이나 기능면에 있어 외국 제품 성능에 미치지 못하고 있다[1].

이는 시장의 한계로 인해 많은 프로그램들이 경쟁할 수 없어 다른 프로그램들에 비해 개발 속도가 현저히 떨어진다. 또한 국내의 스크린리더 개발 업체의 경우, 개발 환경이 열악하고 개발 인원이 적기 때문이다. 이런 개발의 어려움 속에서도 지속적인 업그레이드를 하고 있지만 지원하는 응용프로그램의 업그레이드를 따라가기도 벅찬 것이 현실이다.

좋은 스크린리더란 시각장애인이 보다 다양한 응용 프로그램을 보다 편리하게 사용할 수 있도록 지원하는 것이다. 하지만 적은 개발 인원으로는 다양한 응용 프로그램들을 지원하는 것에 한계가 있다. 예를 들어 메신저를 지원하는 경우 스크린리더 개발 업체에서 지원모듈을 개발하여 발표할 즈음에는 이미 메신저 업체에서 메신저를 업그레이드 하는 경우가 허다하다. 이럴 경우 이미 제작한 메신저 지원 모듈은 쓸모가 없게 되거나, 사용자들이 구버전의 메신저를 사용해야 하는 문제가 발생한다.

이러한 문제를 해결하기 위해서는 개발 속도가 상당히 중요하다. 하지만 다양한 응용 프로그램을 지원해야 하고 그 응용 프로그램이 업그레이드 하자마자

지원 모듈을 발표하기 위해서는 보다 많은 개발 인원이 필수적이다.

하지만 개발 업체에서는 연건상 충분한 인력을 투입할 수 없다. 때문에 시각장애인들이 자주 사용하는 소수의 응용프로그램 만을 지원할 수 밖에 없는 실정이다. 이로 인해 시각장애인들은 자신이 사용하고 싶은 응용프로그램을 마음대로 사용할 수 없고, 스크린리더가 지원하는 특정 응용 프로그램에 한정될 수 밖에 없어 비시각장애인과의 정보 격차가 더욱 벌어질 수 밖에 없다.

본 논문에서는 개발 업체의 인력 부족을 해결하고 보다 다양한 응용 프로그램을 지원할 수 있는 방안으로서 개방형 API 를 제안한다. 본 논문에서 제안한 개방형 API 를 스크린리더 개발 업체에서 채택할 경우 개발 업체는 스크린리더의 주 엔진 모듈과 공통 컨트롤 모듈에 전념할 수 있고, 프로그램이 가능한 사용자나 자원봉사자들이 제안한 API 에 맞도록 플러그인 형태로 구성함으로써 다양한 응용 프로그램을 지원하는 스크린리더 를 보다 빠르게 개발할 수 있을 것이다.

본 논문의 이후 구성은 2 장에서 스크린리더의 구조와 개발 방법에 대해 설명하고, 3 장에서는 개방형 API 를 제안한다. 4 장은 결론으로 구성하였다.

2. 스크린리더 구조 및 개발 방법

스크린리더의 구조는 그림 1 과 같다. 메시지와 이벤트 훅을 통해 정보가 변경된 시점과 윈도우 핸들, 정보 변경 종류와 해당 윈도우의 종류를 알아낸다. 다음으로 윈도우 공통컨트롤[1] 지원 모듈과 응용 프로그램 지원모듈[2]에서 해당 윈도우의 이름, 캡션, 값 등의 정보를 얻어낸다. 이를 변환 및 필터링 과정을 거쳐 유효 정보 텍스트로 바꾸고 음성출력 인터페이스를 거쳐 스피커로 음성을 출력한다[1].

메시지 및 이벤트 처리 부분과 윈도우 공통컨트롤 지원모듈, 변환 및 필터링처리 부분들은 대부분의 응용 프로그램과 컨트롤 윈도우 등에 공통적으로 사용되어 진다. 하지만 응용 프로그램 지원모듈의 경우 각 응용 프로그램의 컨트롤 윈도우마다 모두 다른 방법을 사용한다. 또한 응용프로그램이 하나 개발될 때마다, 또는 응용 프로그램이 업그레이드될 때마다. 다시 작성되어야 한다. 따라서 이 부분이 스크린리더 개발에서 가장 많은 작업량을 가지며, 많은 인력이 필요한 부분이다.

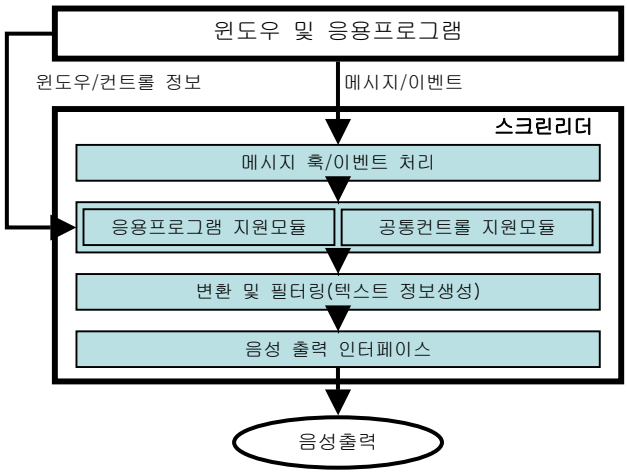

그림 1 스크린리더의 구조

만약 응용 프로그램 지원모듈을 최적화 할 수 있다면 스크린리더 개발의 속도가 빨라질 수 있으며 개발 생산성을 상당이 높일 수 있을 것이다. 본 논문에서는 그 방법으로 개방형 API 를 채택하는 방법을 제안한다. 개발 업체 단독으로는 다양한 응용 프로그램을 지원하는 것이 불가능 하다면, 개방형 API 를 통해 인터페이스를 공개하여 프로그램 가능한 유효 인력을 활용할 수 있도록 하자는 것이다.

그림 2 는 본 논문에서 제안하는 개방형 API 를 채택한 스크린리더의 구조를 보여준다. 기존 방법과 달리 응용 프로그램 지원 모듈을 동적 라이브러리(DLL : Dynamic Link Library)형태로 구성하고, 스크린리더 내부에는 Open API 인터페이스로 대체하였다.

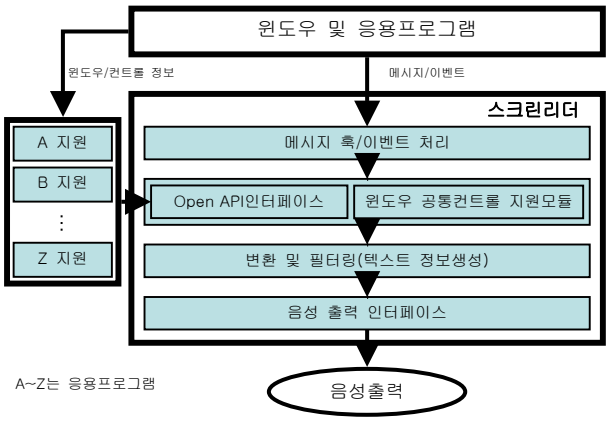

그림 2 동적라이브러리를 채용한 구조

즉 스크린리더는 메시지 및 이벤트 처리 부분과 공통컨트롤 지원모듈, 필터링 및 음성출력 모듈로 구성하고 응용 프로그램 지원모듈은 외부 동적 라이브러리로 만들어 스크린리더에서 사용할 수 있도록 하자는 것이다. 이 때 각 응용 프로그램 지원 모듈은 그림 2 에서 보이는 것처럼 각 응용 프로그램마다 하나의

[1] 공통컨트롤은 버튼, 에디트박스, 체크박스 등 운영체제에서 기본적으로 지원해주는 컨트롤로 본 논문에서는 스크린리더를 위한 기본 모듈인 MSAA[3]를 포함하는 컨트롤로 한정해서 사용한다..

[2] 응용프로그램 지원모듈은 응용 프로그램에 따라 구분하기보다는 윈도우 나 컨트롤의 클래스명에 따라 다른 동작을 하는 경우가 많다. 하지만 윈도우 및 컨트롤이 응용프로그램에 종속된 경우가 많아 본 논문에서는 편의상 응용프로그램 지원 모듈이라는 표현을 사용하고 있다.

동적 라이브러리로 만들어질 수 있으며, 응용 프로그램이 실행될 때 해당 동적 라이브러리를 로드하여 사용하도록 할 수 있다. 이럴 경우 모든 동적 라이브러리를 로드하여 사용할 때보다 메모리 사용 부담도 줄일 수 있을 것이다.

또한 Open API 인터페이스만 공개한다면 누구나 응용 프로그램 지원 모듈을 작성할 수 있게 되어 스크린리더 개발업체의 인력 부담을 줄일 수 있으며, 응용 프로그램 지원 모듈 개발에 사용할 자원을 주엔진 모듈에 투입함으로써 좀 더 우수한 스크린리더를 개발할 수 있을 것이다.

3. 스크린리더 확장을 위한 API 설계

스크린리더에서 공통 컨트롤과 그 특성을 가장 달리하는 부분이 메신저와 에디터 컨트롤이다. 메신저 부분의 경우 수많은 메신저들 마다 다른 인터페이스를 가지고 있기 때문에 스크린리더가 모두를 지원하는 것이 불가능하다. 따라서 국내의 스크린리더의 경우 MSN 메신저만을 지원하고 있는 실정이다.

에디터 컨트롤의 경우에도 HWP, Word, 익스플로러, 파워포인트 등 각 응용 프로그램들 마다 그 특성을 달리 하며, 같은 응용 프로그램이라도 각 버전마다 동작이 다르다.

따라서 이 번 장에서는 다양한 편집기 컨트롤과 메신저를 지원할 수 있는 개방형 API 를 제안하여, 보다 다양한 인력을 활용하여 스크린리더의 기능을 확장할 수 있는 개방형 API 를 설계한다.

3.1 편집기 지원 모듈에 사용한 API

편집기 모듈의 경우 캐럿과 관련한 함수가 정의되어야 한다. 즉 현재 캐럿의 위치에서 이전/현재/다음 글자, 이전/현재/다음 단어, 이전/현재/다음 문장 등을 지원해야 한다. 편집기 모듈의 경우 편집기에만 적용할 수 있는 것은 아니다. 가상 커서를 이용하여 인터넷 익스플로러나 아크로벳 리더 등에서도 활용할 수 있는 기능으로 텍스트 지원 모듈이라고 할 수 있다. 다만 본 논문에서는 이해를 위해 편집기 지원 모듈로 한정하였을 뿐이다.

편집기 모듈에서 정의한 API 는 표 1 과 같다. 먼저 해당 윈도우의 텍스트 정보를 가져오는 GetText 함수를 정의 하였다. GetText 함수는 윈도우 핸들과 구분 파라메터를 가지며 그 값에 따라서 글자/단어/줄/문단 등의 텍스트를 가져온다. 이때 가져오는 텍스트는 캐럿의 앞이나 뒤 또는 바로 캐럿위치의 글자를 지정할 수 있다.

다음으로 현재 캐럿이 있는 곳의 문단 정보와 글자 정보에 관련한 함수도 정의하였다. GetFontInfo 의 경우 현재 캐럿위치의 글자 모양에 대한 정보를 보내준다.

GetParaInfo 의 경우 현재 캐럿 위치의 문단정보를 알려준다.

또한 워드나 HWP 의 경우 표를 많이 사용한다. 때문에 표에 대한 정보를 알려주는 것은 스크린리더의 필수 기능중에 하나가 되어있다. 따라서 표 정보에 관련한 함수들도 정의 하였다. GetCellPos 함수의 경우 현재 셀이 테이블 내에 어느 위치에 있는지를 검사하고, 현재 행과 열을 리턴한다.

다음으로 GetTableCaption 은 표의 제목을 알려주고, GetCellHeader 는 현재 셀의 제목을, GetCellText 는 현재 셀의 내용을 알려준다.

표 1 편집기를 위한 API 함수

int GetText(HWND win, char *text, int mode) ;
- mode 에 따라서 이전/현재/다음의 글자/단어/문장 등리턴한다.

int GetFontInfo(HWND win, char *fontname, int h, int w) ;
- 현재 캐럿 위치의 폰트 정보를 리턴한다.

int GetParaInfo(HWND win , TParaInfo *parainfo) ;
- 현재 캐럿 위치의 문단정보를 리턴한다.

int GetCellPos(HWND win, int *row, int *col) ;
- 테이블에서 위치정보

int GetCelleText(HWND win, char *text) ;
- 현재 셀의 텍스트를 가져온다.

int GetTableCaption(HWND win, char *text) ;
- 표 제목을 가져온다.

int GetCellHeader(HWND win, char *row, char *col) ;
- 현재 셀의 행제목과 열제목을 가져온다

int GetCellInfo(HWND win, int row, int col, TInfo info);
- row, col 위치 셀의 정보를 가져온다.

3.2 메신저 모듈에 사용할 API

메신저의 경우도 시각장애인들이 많이 사용하는 응용프로그램이지만 잦은 업그레이드로 사용에 제한을 받는 경우가 많다. 따라서 메신저를 지원할 모듈 인터페스를 개방하여 사용자가 직접 작성하여 사용하게 하므로써 원하는 어떤 메신저도 사용할 수 있도록 했으며, 메신저의 잦은 업그레이드에도 쉽게 대처할 수 있을 것이다.

먼저 메신저의 경우 지금까지 입력된 글자를 가져오는 함수가 필요하다. 이는 상대방이 입력한 내용을 스크린리더에서 읽어줘야 하기 때문에 지금까지 들어온 문장을 버퍼에 저장하고 있어야 한다. GetHistoryBuffer 함수는 바로 이러한 기능을 수행한다. 보통 이러한 기능은 메신저의 히스토리 윈도우를 통해 가져올 수 있다.

다음으로 새로 들어온 문장만을 알려주는 GetNewMessage 함수를 정의하였다. GetNewMessag 함수가 호출 된 이후 새롭게 들어온 메시지들을 알려준다. GetNewMessage 의 경우에는 사용자명을 알려주는 옵

션과 사용자명을 알려주지 않는 옵션을 따로 두었다. 즉 옵션에 따라서 메시지를 파싱하고 사용자명을 함께 리턴하거나 사용자명을 제거하고 리턴할 수 있도록 해야 한다. 이는 두 사람이 채팅할 경우 사용자명을 알려주는 것은 불필요한 정보를 주는 것이기 때문에 일반적인 스크린리더의 경우 이를 토글키를 이용하여 선택할 수 있도록 하고 있다. 따라서 GetNewMessage 함수에도 이러한 옵션은 필수적이다.

다음으로 방금 전에 들어온 2,3 개의 메시지를 알려주는 함수가 정의되어야 한다. 이는 GetNewMesasge 로 읽는 도중 다른 간섭에 의해 메시지를 잃어버리는 경우가 빈번하기 때문이다. 따라서 이를 위해서 방금 전에 들어왔던 메시지를 다시 알고 싶은 경우 이를 활용할 수 있다 .

해당 함수의 형태는 표 2 에 나열하였다..

표 2 메신저를 위한 API 함수

int GetHistoryBuffer(HWND win, char *text) ;
- 현재까지 전체 내용을 가져온다.

int GetNewMessage(HWND win, char *text, int opt) ;
- 새로 들어온 메시지를 가져온다

int GetLastMessage(HWND win , TParaInfo *parainfo) ;
- 최근에 들어온 몇 개의 메시지를 가져온다.

4. 결론

지금까지 스크린리더 개발의 생산성을 향상시킬 수 있는 개방형 인터페이스를 제안하였다. 제안한 방법을 사용할 경우 보다 많은 인력을 활용할 수 있어 개발업체는 스크린리더의 성능 향상에만 전념하고, 프로그램이 가능한 자원 봉사자나 사용자 등의 인력을 활용하여 각 응용 프로그램션을 지원할 수 있는 모듈을 작성함으로써 보다 효율적인 개발이 가능할 것이다. 또한 다양한 응용 프로그램을 지원함으로써 시각장애인들의 정보 접근성을 향상시킬 수 있으며 이를 통해 시각장애인들도 보다 많은 정보를 얻을 수 있는 기회를 가질 것으로 기대한다.

지금까지 각 응용프로그램을 지원하는 방안을 제시하였다. 하지만 보다 중요한 것은 응용프로그램 개발 방법이다. 응용 프로그램을 설계 시점부터 정보 접근이 어렵지 않도록 MSAA 에 충실하도록 프로그램을 설계한다면, 현재 나와 있는 스크린리더의 기능만으로도 시각장애인들이 응용프로그램을 쉽게 사용할 수 있을 것이다.

참고문헌

[1]이승수,윈도우 스크린리더의 기술적 고찰, 제 11 회 정보과학회 인간과 컴퓨터 상호작용 연구회 학술대회 워크샵자료집, 2002. 2

[2] 이승수, 민경석, 주용덕, 강성찬, 김석일, "시각장애인을 위한 인터넷 솔루션 개발", 정보과학회 2000 춘계학술발표회논문집, 제 27 권, 제 1 호(B), pp.405-407(2000.4)

[3] http://www.microsoft.com/enable/

[4] http://www.gwmicro.com/wemanual

[5]http://www.freedomscientific.com/fs_products/software_jaws.asp