**Министерство образования Украины**

**Международный Соломонов Университет**

**кафедра програмного обеспечения автоматизированных систем**

**Курсовая работа**

**по курсу "Основы теории алгоритмов"**

**"Проектирование алгоритмов чтения данных с экрана"**

**студента 4-го курса Факультета Компьютерных Наук**

**Прохорова Андрея Витальевича**

**Руководитель проекта Терзян Т. К.**

Киев 1998 р.Введение

Можно много спорить о вредном влиянии персонального компьютера на здоровье, психику и жизнь людей вообще, однако бесспорен тот факт, что среди своих недостатков компьютер имеет намного больше достоинств. В частности, это невероятно универсальный инструмент, который с успехом используется практически во всех сферах человеческой деятельности, включая важнейшие: медицину, образование, военную промышленность, искусство, математические науки.

Для многих персональный компьютер является средством выражения своего творческого потенциала, своего “я”. Так, художник может легко творить компьютерные шедевры, писатель может быстро написать свою книгу, математик может рассчитать задачу без утомительных расчетов сложных формул на бумаге. А для людей с определенными физическими недостатками компьютер представляет собой вообще чуть ли не единственное средство для выражения своего “я”, своего интеллекта и творческого потенциала, особенно это касается людей незрячих и лишенных возможности передвигаться. Например, стоит упомянуть одного из величайших физиков современности – Стива Хокинга. Этот человек, заболевший в детстве, практически утратил возможность двигаться и говорить. Однако с помощью миниатюрного персонального компьютера и специальной программы ученый имеет возможности писать работы и проводить лекции, неизменно пользующиеся успехом.

Представленный здесь проект является еще одной попыткой облегчить жизнь незрячим людям. Разработанная программа представляет собой так называемый screen-reader (букв. “читатель с экрана”), который адаптирован для использования с системой электронных словарей Polyglossum-II для DOS. Программа позволяет озвучивать названия и содержание словарных статей в окне программы-словаря. Таким образом, screen-reader позволит незрячим людям заниматься переводом научных и других работ с одного языка на другой.

***Постановка задачи***

Необходимо разработать алгоритм озвучивания процесса работы с резидентными программами: чтение информации, выводимой на экран, озвучивание команд управления, введение дополнительных команд. Алгоритм необходимо реализовать на примере программы screen-reader, которая позволит озвучить названия и содержание словарных статей при просмотре их в системе электронных словарей Polyglossum-II для MS-DOS.

**Глава 1**

**Анализ алгоритмов чтения данных с экрана монитора**

***Анализ проблем***

Разработка программы screen-reader связана с решением следующих проблем:

а) взаимодействие программы-ридера с пользователем;

б) взаимодействие программы-ридера с озвучиваемой программой;

в) соблюдение соглашений операционной среды при одновременной работе нескольких программ.

Взаимодействие программы-ридера с пользователем

Интерфейс пользователя в программах, которые необходимо адаптировать для использования людьми незрячими, приходится менять. Ведь те особенности работы программы, которые доступны и очевидны зрячему пользователю, вовсе не очевидны и порой недоступны для человека незрячего, между тем последний не должен терять возможности нормального взаимодействия с программой, как это может делать обычный человек. В силу этих требований, при разработке программ такого рода неизменно возникает необходимость “ломать” традиционную структуру взаимодействия программы с пользователем, менять интерфейс пользователя, приспосабливать для незрячих людей. Приходится добавлять новые режимы работы программы, новые команды пользователя, изменять поведение озвучиваемой программы.

Взаимодействие программы-ридера с озвучиваемой программой

Существуют разнообразные программы-синтезаторы речи. Принимая решение о выборе той или иной программы-синтезатора, необходимо четко понимать возможности, которые каждый синтезатор может предоставить, а также то, как работает сама программа, какие ограничения она накладывает в работе при взаимодействии с другими программами и окружением операционной среды, какие требования предъявляются к ресурсам компьютера и т.д. Очевидным представляется необходимость знать соглашения интерфейса программы-синтезатора, необходимо знать формат данных, которыми оперирует синтезатор.

Выяснение таких подробностей позволит оптимальным образом подобрать наиболее подходящую программу для конкретной задачи озвучивания и избежать потенциальных проблем при разработке озвучивающих программ.

Соблюдение соглашений операционной среды при одновременной работе нескольких программ.

Важным моментом при разработке озвучивающих программ является знание особенностей работы операционной среды, в которой работают непосредственно озвучиваемые программы, синтезаторы речи. Это позволит быстро и эффективно наладить взаимодействие комплекса программ, необходимых для озвучивания данных, а также позволит существенно улучшить качество работы такого комплекса и сократить время, необходимое на отладку программ в процессе разработки.

**Глава 2**

**Проектирование алгоритма озвученного управления системой электронных словарей "Polyglossum-II"**

Первым делом при разработке программы-ридера для работы с электронными словарями возникла необходимость продумать новый интерфейс пользователя, адаптированный для незрячего человека. Используя исходную систему команд программы-словаря, обычный человек всю отображаемую информацию мог легко замечать и воспринимать. При этом если на экран выводилось одновременно несколько строк текста, человек мог охватить взглядом весь текст. Естественно, человек, лишенный зрения, нуждается в дополнительной системе команд, которая позволила бы ему прослушать каждую строчку, выведенную на экране, а при необходимости вывести на экран и озвучить остальную информацию, не отображенную в настоящий момент на дисплее. Поэтому, например, при просмотре содержимого словарной статьи, программа-ридер должна переходить в новый режим работы, при котором по нажатию курсорных клавиш можно было не листать всю статью вместе с ее содержимым на экране, а озвучить каждую отдельную строчку. Таким образом, имеет место изменение структуры интерфейса пользователя.

В процессе разработки возникли некоторые проблемы, связанные непосредственно с интеграцией программы-ридера с системой словарей. Не имея ни исходного текста программ, ни данных о формате самих словарей, было проблематичным найти оптимальное решение для обеспечения взаимодействия программы-ридера и программой-словарем.

Система электронных словарей, а также использованный синтезатор речи рассчитаны на применение исключительно под ОС MS-DOS. Вследствие этого, при разработке резидентной программы были учтены соответствующие соглашения данной операционной системы при работе с прерываниями, была учтена особенность MS-DOS, связанная с повторной входимостью (реентерабельностью). Пренебрежение подобными особенностями функционирования ОС могут привести к нестабильной работе программ или вообще не позволит реализовать поставленные цели.

Для решения поставленной задачи был предложен следующий алгоритм: программа-ридер, являясь резидентной, перехватывает прерывание клавиатуры, записывая себя в таблицу векторов на место оригинального обработчика прерывания, предварительно сохранив адрес последнего. При нажатии на любую клавишу инициируется резидентная программа-ридер. Выяснив, активна ли в настоящий момент программа-словарь, программа анализирует код нажатой клавиши. Если код не принадлежит группе кодов, обрабатываемых программой, управление передается оригинальному обработчику прерывания. Если же нажатая клавиша принадлежит системе команд обрабатываемых программой-ридером, то происходит считывание выведенной в определенное место экрана информации. Преобразовав считанную информацию в специальный формат, ридер вызывает резидентную программу-синтезатор речи (в проекте использовалась сторонняя программа-синтезатор “Фонемафон”), после чего передает управление оригинальному обработчику прерывания. Предложенный алгоритм представляет собой оптимальное решение поставленной задачи, обеспечивая при этом относительную простоту реализации и функциональность программы. При необходимости использовать программу-ридер в других озвучиваемых системах, в большинстве случаев не будет надобности существенно менять алгоритм для адаптации к новой среде.

Схема “Перехват нажатия клавиш” ::==

ПОКА НЕ ‘Нажата клавиша ESC’

ЦИКЛ (“Ждать нажатия клавиш” \*

\* Выбор (

‘Клавиша ’  “Получить координаты курсора” \*

\* ПОКА НЕ ‘Прочитаны все символы’

ЦИКЛ (“Читать очередной символ с экрана в буфер”) \*

\* “Преобразовать содержимое буфера в новый формат” \* “Произнести строку”;

‘Клавиша ’  “Получить координаты курсора” \*

\* ПОКА НЕ ‘Прочитаны все символы’

ЦИКЛ (“Читать очередной символ с экрана в буфер”) \*

\* “Преобразовать содержимое буфера в новый формат” \* “Произнести строку”;

‘Клавиша ENTER’ **“**Переместить курсор на первую строку словарной статьи” \*

\* ПОКА НЕ ‘Прочитаны все символы’

ЦИКЛ (“Читать очередной символ с экрана в буфер”) \*

\* “Преобразовать содержимое буфера в новый формат” \* “Произнести строку” \*

\* ПОКА НЕ ‘Нажата клавиша ESC’

ЦИКЛ (“Ждать нажатия клавиши” \*

Выбор (

‘Клавиша ’  “Получить координаты курсора” \*

\* ПОКА НЕ ‘Прочитаны все символы’

ЦИКЛ (“Читать очередной символ с экрана в буфер”) \*

\* “Преобразовать содержимое буфера в новый формат” \* “Произнести строку”;

‘Клавиша ’  “Получить координаты курсора” \*

\* ПОКА НЕ ‘Прочитаны все символы’

ЦИКЛ (“Читать очередной символ с экрана в буфер”) \*

\* “Преобразовать содержимое буфера в новый формат” \* “Произнести строку”;

‘Клавиша ENTER’ **“**Переместить курсор на первую строку словарной статьи” \*

\* ПОКА НЕ ‘Прочитаны все символы’

ЦИКЛ (“Читать символ с экрана в буфер”) \*

\* “Преобразовать содержимое буфера в новый формат” \* “Произнести строку” \*

\* ПОКА НЕ ‘Нажата клавиша ESC’

ЦИКЛ (“Ждать нажатия клавиши” \*

\* Выбор (

‘Клавиша ’  “Получить координаты курсора” \*

\* ПОКА НЕ ‘Прочитаны все символы’

ЦИКЛ (“Читать очередной символ с экрана в буфер”) \*

\* “Преобразовать содержимое буфера в новый формат” \* “Произнести строку”;

‘Клавиша ’  “Получить координаты курсора” \*

\* ПОКА НЕ ‘Прочитаны все символы’

ЦИКЛ (“Читать очередной символ с экрана в буфер”) \*

\* “Преобразовать содержимое буфера в новый формат” \* “Произнести

строку”; ))))));

**Глава 3**

**Реализация и апробация программы**

Результаты экспериментов

Программа-ридер была одновременно реализована на двух языках программирования – языке Pascal и языке Ассемблера. Такое решение обосновано тем, что реализация программы-ридера на языке Pascal не требует больших усилий в силу его гибкости. Кроме того, легкость программирования на этом языке существенно облегчит дальнейшее изменение алгоритма чтения данных с экрана для адаптации программы-ридера к работе в новой среде. С другой стороны, язык Ассемблера как никакой другой язык подходит для решения системных задач, что актуально при создании резидентных программ, работающих с прерываниями и с учетом соглашений операционной среды. Теоретически, язык Ассемблера при грамотной реализации программы поможет избежать проблем, которые могут иметь место при написании подобных программ с использованием языков высокого уровня.

На этапе разработки программы было проведено множество экспериментов и других действий, направленных на выяснение внутренних принципов работы программы-словаря и способов интеграции резидентной программы-ридера непосредственно со словарями. Установлено, что программа-словарь состоит из двух резидентных программ. Первая программа полностью написана на языке Ассемблера. Перехватив нажатия горячих клавиш, вызывающих словари, программа активизировала вторую резидентную программу, которая управляла просмотром статей в словарях. Вторая программа комплекса была написана на языке Pascal. Такое замысловатое взаимодействие несколько затруднило на первых этапах реализацию программы-ридера. После многочисленных попыток и проб было принято решение написать резидентную программу-ридер, алгоритм и принцип функционирования которой описан выше. Было также выяснено, что лишь определенный порядок загрузки программ (программы-словаря, программы синтеза речи, резидентной программы-ридера) обеспечивает должное функционирование комплекса программ в целом.

Никакие другие предпринятые методы реализации чтения текста с экрана с последующим произношением не увенчались успехом в силу специфики работы резидентной программы-словаря. В таких случаях система либо мгновенно зависала, либо функционировала крайне нестабильно, что, естественно, не позволяло нормально работать со словарем. Поэтому предложенный метод выглядит как наиболее оптимальный.

Еще одна проблема, которая возникла в процессе реализации screen-reader’a – это неспособность использованного синтезатора речи на приемлемом уровне качества воспроизводить текст на английском языке. Такая проблема решаема только за счет применения альтернативного синтезатора речи, который предоставляет такие возможности. Можно использовать два независимых синтезатора речи, один для произношение слов на русском языке, другой синтезатор – для произношения речи на английском языке. Однако такой подход накладывает новые ограничения на ресурсы компьютера.

**Заключение**

Целесообразность реализации данной программы не вызывает никаких сомнений. Польза, принесенная от использования данной программы и вообще программ такого класса незрячими людьми, очевидна. Эти люди имеют возможность принести пользу обществу, реализовать себя в области переводов материалов с одного языка на другой.

Приложение 1.

Листинг исходного текста программы-ридера

{$M $800,0,0}

program test;

uses Crt, Dos;

var

Reg,OldReg,load : registers;

flag:boolean;

OldInt09:pointer;

Old09 : procedure;

KurX,KurY: integer;

OldX,OldY: integer;

aaa:array[1..2] of word absolute OldInt09;

S: string[80];

q : string;

x0,y0 ,

x1,y1 ,

x2,y2,j : integer;

var fl1:byte;

type

TScr = array[1..25\*80, 1..2] of char;

TaScr = ^TScr;

var

Scr0 : TASCR;

var

SlovYes : boolean;

SlInByte : byte;

SlX,SlY,SlX0,SlY0 : integer;

Sl0Key : boolean;

procedure SText (Sent00 : string

var REG : registers;

L : integer;

begin

Sent00 := copy (Sent00, 1, 250);

REG.es:=Seg(Sent00);

REG.SI := Ofs(Sent00) +1;

L:=length(Sent00) +1;

REG.CX:=L;

Sent00 [Reg.CX]:=',';

reg.AX:=256;

Intr($80,REG);

end;

procedure New09; interrupt;

var i,l,r:integer;

begin

SlInByte:=0;

if Port[$60] >= $80

then begin

SlInByte:=Port[$60]-$80;

if (SlInByte=28)

then begin

sl0key:=true;

end

end;

if (SlInByte in [63,72,80]) and (SCR0^[410,1]='Й')

then

begin

if Sl0key

then

begin

SlX:=11{SlX0}; SlY:=7{SlY0};

gotoxy(slx,sly);

sl0key:=false;

end

else

begin

SlX:=WhereX;

SlY:=WhereY;

end;

l:=80\*(sly-1)+SlX;

r:=l;

if (sly>0) and (sly<20)

then

begin

while (l > 80\*(sly-1) ) and not(SCR0^[l,1] in ['°'..'Я']) do dec(l);

while (r < 80\*sly) and not(SCR0^[r,1] in ['°'..'Я']) do inc(r);

dec(r);

while (r<l) and (SCR0^[r,1]=' ') do dec(r);

s:='';

for i:=l+1 to r

do s:=s+SCR0^[i,1];

stext(s);

SlX0:=SlX; SlY0:=SlY;

end;

end;

old09;

end;

begin (\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*)

GetIntVec($9,@Old09);

SetIntVec($9,Addr(New09));

SCR0:=ptr($b800,$0000);

KEEP(0);

end.