(19) **Bundesrepublik Deutschland**
Deutsches Patent- und Markenamt

(10) **DE 696 29 120 T2** 2004.04.22

(12) # Übersetzung der europäischen Patentschrift

(97) **EP 0 852 047 B1**
(21) Deutsches Aktenzeichen: **696 29 120.7**
(86) PCT-Aktenzeichen: **PCT/AU96/00552**
(96) Europäisches Aktenzeichen: **96 928 282.1**
(87) PCT-Veröffentlichungs-Nr.: **WO 97/009704**
(86) PCT-Anmeldetag: **04.09.1996**
(87) Veröffentlichungstag der PCT-Anmeldung: **13.03.1997**
(97) Erstveröffentlichung durch das EPA: **08.07.1998**
(97) Veröffentlichungstag der Patenterteilung beim EPA: **16.07.2003**
(47) Veröffentlichungstag im Patentblatt: **22.04.2004**

(51) Int Cl.[7]: **G09B 17/04**
G06F 17/21, G09B 5/06

(30) Unionspriorität:
PN520495 04.09.1995 AU

(84) Benannte Vertragsstaaten:
AT, BE, CH, DE, DK, ES, FI, FR, GB, GR, IE, IT, LI, LU, MC, NL, PT, SE

(73) Patentinhaber:
Charon Holdings Pty. Ltd., Townsville, Queensland, AU

(72) Erfinder:
EDGAR, Andrew, Mark, Townsville, AU

(74) Vertreter:
Patentanwälte Reinhardt & Pohlmann Partnerschaft, 75172 Pforzheim

(54) Bezeichnung: **LESEHILFE**

Anmerkung: Innerhalb von neun Monaten nach der Bekanntmachung des Hinweises auf die Erteilung des europäischen Patents kann jedermann beim Europäischen Patentamt gegen das erteilte europäische Patent Einspruch einlegen. Der Einspruch ist schriftlich einzureichen und zu begründen. Er gilt erst als eingelegt, wenn die Einspruchsgebühr entrichtet worden ist (Art. 99 (1) Europäisches Patentübereinkommen).

Die Übersetzung ist gemäß Artikel II § 3 Abs. 1 IntPatÜG 1991 vom Patentinhaber eingereicht worden. Sie wurde vom Deutschen Patent- und Markenamt inhaltlich nicht geprüft.

Beschreibung

Technisches Gebiet

[0001] Die vorliegende Erfindung bezieht sich auf eine Lesehilfe und insbesondere auf ein System und ein Verfahren zur Darstellung von Text- und/oder graphischen Informationen zum Lesen. Derartige Lesehilfen sind z. B. aus US 3,938,139 bekannt.

Stand der Technik

[0002] Über die Jahre sind viele verschiedene Techniken zur Erhöhung der Lesegeschwindigkeit und zur Aufrechterhaltung des Verständnisses sowie zum Rückbehalt des Lesestoffes entwickelt und gefördert worden. Diese Techniken sind damit einher gegangen, sich vom traditionellen Verfahren des Lesens eines Textes zu lösen, der in gedrucktem Seitenformat dargestellt ist, wobei jedes aufeinander folgende Wort in Folge über die Seite hinweg und Zeile für Zeile gelesen wird. Der Grund für die Entwicklung der verschiedenen Lesetechniken ist weitestgehend bestimmten Nachteilen zugewiesen worden, die mit der traditionellen Lesemethode verbunden sind, die mindernd entgegen einer Erhöhung der Lesegeschwindigkeit und einer Aufrechterhaltung des Verständnisses wirkt. Vielmehr kann die Herangehensweise an eine vollständig bedruckte Seite zur Darstellung von Text, bei der der Text grundsätzlich statisch ist und der Leser gefordert ist sein Auge entlang einer Zeile zu bewegen, eine Ablenkung oder Verwirrung des Lesers dadurch verursachen, dass der gesamte Text sowohl oberhalb, unterhalb als auch neben dem zu lesenden Wort erscheint. Als Folge davon leiden viele Leser darunter, dass sie Zeilen des Textes, Sätze und Paragraphen wiederholt lesen müssen, was die Leserate verringert und der Aufrechterhaltung eines Verständnisses und einem Behalten des Lesestoffes während der Zeit entgegen wirkt, in der der Text das erste Mal gelesen wird.

[0003] Alternative Lesetechniken haben versucht, dieses spezielle Problem dadurch anzusprechen, dass der Leser vom alten Lesestil weg trainiert wurde, indem Maschinen verwendet wurden, die jene Abschnitte der Seite ausblenden, die nicht zu einem bestimmten Zeitpunkt gelesen werden und/oder alternativ durch ein Hervorheben jenes Abschnittes des Textes, der gelesen wird. Ein Nachteil dieser Techniken besteht darin, dass der Text immer noch passiv bleibt, da das Auge des Lesers immer noch sich von Wort zu Wort bewegen muss und mit dem hervorgehobenen Abschnitt des Textes Schritt halten muss. Ferner sind mit Annahme dieser Techniken diese sehr maschinenabhängig und waren somit vor dem modernen Computerzeitalter aufwändig zu implementieren.

[0004] Mit der Einführung und weit verbreiteten Akzeptanz und Gebrauch heutiger Computersysteme können derartige Lesetechniken zu einem Bruchteil der Kosten implementiert werden, die früher erforderlich waren. Zudem können mit der Leistung moderner Computersysteme hochentwickeltere und verbesserte Lesetechniken entwickelt und benutzt werden, um die Lesegeschwindigkeit zu verbessern und das Verständnis und das Behalten des Lesestoffes auf einer Vollzeitbasis beizubehalten, wobei Text gespeichert und auf einem Computerterminal angezeigt wird im Gegensatz zur einfachen Nutzung als Trainingswerkzeug.

[0005] Mit der Popularität von Informationsnetzwerken und Datenbanken wie dem Internet und der Nutzung von elektronischer Post als auch Massenmedien wie der CDROM zur Übertragung und Speicherung großer Mengen an Textinformationen besteht jetzt eine viel größere Veranlassung für Personen, ihre Lesegeschwindigkeit und ihr Verständnis von Textinformation zu verbessern, als dies zuvor jemals der Fall gewesen ist. Kombiniert man dies mit der inhärenten Leistung moderner Personal-Computer und dem visuellen Dynamismus von graphischen User Interfaces wie WINDOWS (registrierte Marke) und anderen Multitasking-Fensterumgebungen besteht eine ideale Gelegenheit anspruchsvollere Lesetechniken einzuführen und diese permanent mit geeigneter Anwendungssoftware einzubinden, die für die Behandlung derartiger Textinformationen eingesetzt wird, um eine neue Umgebung zum Lesen bereit zu stellen, die sich selbst Hilfe zum schnelleren Umgang und zum verbesserten Verständnis leistet.

Offenbarung der Erfindung

[0006] Es ist daher eine Aufgabe der vorliegenden Erfindung ein System und ein Verfahren zur Darstellung von Text- und/oder graphischer Information zum Lesen in einer modernen Computerumgebung zur Verfügung zu stellen, die es einen Benutzer davon ermöglichen, seine Lesegeschwindigkeit zu erhöhen und das Verständnis und die Merkfähigkeit des Lesestoffes beizubehalten oder zu verbessern.

[0007] Gemäß einem Aspekt der vorliegenden Erfindung wird ein System zur Darstellung von Text- und/oder graphischen Informationen zum Lesen geschaffen, umfassend:

Anzeigemittel zum Anzeigen der Text- und/oder graphischen Informationen auf einem Anzeigemedium;

Verarbeitungsmittel zum Empfangen der Text- und/oder graphischen Informationen in einer elektronisch codierten Form und zur Verarbeitung dieser zur Präsentation auf dem Anzeigemedium;

Steuermittel zum Steuern der Präsentation der Text- und/oder graphischen Informationen auf dem Anzeigemedium; und
Eingabemittel zum Eingeben von Steuersignalen zum Betrieb des Verarbeitungsmittels und des Steuermittels oder zum Verändern von Steuerparametern für das Verarbeitungsmittel und das Steuermittel;
wobei das Steuermittel eine Reihe von Steuertunktionen umfasst, die aufgerufen werden können, um sequentiell diskrete und sukzessive Teile der Text- und/oder graphischen Informationen mit einem vorbestimmten Ausblendzeitabstand an dieselbe Position auf dem Anzeigemedium zu bringen, wobei jeder diskrete und sukzessive Teil eine vorbestimmte Zeit lang an dieser Position angezeigt wird; und
gekennzeichnet durch eine der Steuerfunktionen, die ein Zentrierungsmittel zum lateralen Zentrieren jedes der Teile in der Position enthält.

[0008] Vorzugsweise enthält das Steuermittel Anwendungsmittel zum Definieren des allgemeinen Layouts des Anzeigemediums für die Präsentation der Text- und/oder graphischen Informationen und der visuellen Steuerattribute des Systems.

[0009] Vorzugsweise reagiert das Anwendungsmittel direkt auf die Steuerparameter für das Steuermittel und umfasst bestimmte Steuerfunktionen zum Definieren und Spezifizieren bestimmter Eigenschaften der Präsentation der Text- und/oder graphischen Informationen in dem allgemeinen Layout gemäß der Einstellung eines bestimmten Steuerparameters dafür und der Eingabe der Steuersignale aus dem Eingabemittel.

[0010] Vorzugsweise umfassen die Steuermittel Ausschnittmittel zum Anordnen des allgemeinen Layouts in einem oder mehreren Ausschnitten, in denen die Text- und/oder graphischen Informationen präsentiert werden.

[0011] Die Ausschnitte umfassen vorzugsweise einen Flash-Ausschnitt und einen Browser-Ausschnitt, wobei der Flash-Ausschnitt die Position und der Browser-Ausschnitt einen Bereich enthält, in dem ein größerer Teil der Text- und/oder graphischen Informationen, aus dem die diskreten Teile abgeleitet werden, angeordnet werden kann.

[0012] Die Steuermittel umfassen vorzugsweise Zentrierungsscrollmittel zum automatischen Scrollen der Präsentation des größeren Teils in dem Browser-Ausschnitt und zum Zentrieren der Zeile des größeren Teils, der die entsprechenden Text- und/oder graphischen Informationen für den diskreten Teil präsentiert, in dem Browser-Ausschnitt.

[0013] Die Ausschnittmittel umfassen vorzugsweise Hervorhebemittel zum Hervorheben der entsprechenden Text- und/oder graphischen Informationen in dem Browser-Ausschnitt.

[0014] Vorzugsweise umfassen die Steuermittel Verzögerungsmittel zum Einstellen der Steuerparameter für die vorbestimmte Ausblendezeit und die vorbestimmte Anzeigezeit.

[0015] Die Steuermittel umfassen vorzugsweise Teilauswahlmittel zum Einstellen der Steuerparameter für den diskreten Teil oder Abschnitt.

[0016] Die Steuermittel zeigen vorzugsweise jeden diskreten und sukzessiven Teil oder Abschnitt der Text und/oder graphischen Informationen mit einer vorgeschriebenen Ausblendezeit voneinander getrennt an.

[0017] Das Verarbeitungsmittel ist vorzugsweise in Mikrocomputer mit einer graphischen Benutzeroberfläche mit einer Multitasking-Fensterbetriebsfunktion.

[0018] Vorzugsweise umfasst das Steuermittel ein Zeitmanagementmittel zum Überwachen der voraussichtlichen und vergangenen Zeit von dadurch ausgeführten Funktionen und zum Koordinieren der Abgabe und Rückgabe der Steuerung zu und von der Multitasking-Fensterbetriebsfunktion.

[0019] Nach einem anderen Aspekt der vorliegenden Erfindung wird ein Verfahren zum Darstellen von Text- und/oder graphischen Informationen zum Lesen geschaffen, mit den folgenden Schritten:
Empfangen der Text- und/oder graphischen Informationen in einer elektronisch codierten Form;
Verarbeiten dieser zur Präsentation auf einem Anzeigemedium; Auswählen diskreter und sukzessiver Teile der Text- und/oder graphischen Informationen;
sequentielles Anzeigen der Teile an derselben Position des Anzeigemediums mit einem vorbestimmten Ausblendzeitabstand;
Anzeigen jedes diskreten und sukzessiven Teils an der Position für eine vorbestimmte Anzeigezeit; und
laterales Zentrieren jedes der Teile in der Position.

Kurzbeschreibung der Zeichnungen

[0020] Die Erfindung ist im Licht der folgenden Beschreibung anhand eines bestimmten Ausführungsbeispiels zu verstehen. Die Beschreibung erfolgt unter Bezug auf die beigefügten Bezeichnungen, wobei:

[0021] **Fig.** 1 ein Blockdiagramm der funktionalen Hauptkomponenten ist, die das System ausmachen;

[0022] **Fig.** 2 ein Blockdiagramm ist, das die Klassen von Funktionen darstellt, die die Steuermittel ausmachen;

[0023] **Fig.** 3 ein Ablaufdiagramm der verschiedenen durch das System ausgeführten Funktionen ist;

[0024] **Fig.** 4 eine Kopie des allgemeinen Bildschirmlayouts des Hauptbildschirmsist;

[0025] **Fig.** 5 eine Kopie des Bildschirmlayouts ist, die das Pull-down-Menü ‚File' zeigt;
[0026] **Fig.** 6 eine Kopie des Bildschirmlayouts ist, wobei die Dialogbox ‚Open' offen ist und bereit ist, um eine Textdatei zum Lesen zu öffnen;
[0027] **Fig.** 7 eine Kopie eines Bildschirmlayouts ist, das den Browserausschnitt und den Flash-Ausschnitt zeigt, wobei der erste Teil der zu lesenden Textdatei im Browserausschnitt dargestellt ist;
[0028] **Fig.** 8 eine mit **Fig.** 7 ähnliche Darstellung zeigt, wobei jedoch das Pull-down-Menü, Options' gezeigt ist;
[0029] **Fig.** 9 eine Kopie eines Bildschirmlayouts zeigt, dass die Dialogbox ‚Delays' offen zeigt, nachdem dieselbe gemäß **Fig.** 8 ausgewählt wurde;
[0030] **Fig.** 10 eine Kopie eines Bildschirmlayouts ist, bei der die Dialogbox ‚Font' offen gezeigt ist, nachdem diese aus dem Menü ‚Options', wie in **Fig.** 8 dargestellt, ausgewählt wurde;
[0031] **Fig.** 11 eine Kopie eines Bildschirmlayouts zeigt, bei dem die Dialogbox ‚Flashes' offen ist, wenn sie aus dem Menü, Options, wie in **Fig.** 8 gezeigt, ausgewählt wurde;
[0032] **Fig.** 12 eine Kopie eines Bildschirmlayouts ist, das nur den Flash-Ausschnitt anzeigt, wobei ein einzelnes Wort dargestellt ist;
[0033] **Fig.** 13 eine ähnliche Darstellung zu **Fig.** 12 zeigt, wobei jedoch ein anderes Wort angezeigt ist, das jedoch ähnlich lateral zentriert ist;
[0034] **Fig.** 14 eine ähnliche Darstellung zu **Fig.** 7 zeigt, wobei jedoch das Pull-down-Menü, Read' gezeigt ist;
[0035] **Fig.** 15 eine Kopie eines Bildschirmlayouts ist, bei dem sowohl der Browser-Ausschnitt als auch der Flash-Ausschnitt gezeigt sind, wobei ein Textblock innerhalb des Browser-Ausschnitts ausgewählt ist;
[0036] **Fig.** 16 eine ähnliche Darstellung zu **Fig.** 7 zeigt, wobei jedoch das Pull-down-Menü, Speed' gezeigt ist;
[0037] **Fig.** 17 eine ähnliche Darstellung zu **Fig.** 7 zeigt, wobei der Beginn des Leseprogramms gezeigt ist, wobei der Flash-Ausschnitt einen diskreten Teil des zu lesenden Textes präsentiert und der Browser-Abschnitt den entsprechenden Abschnitt des Textes hervorhebt; und
[0038] **Fig.** 18 eine andere Darstellung des allgemeinen Bildschirmlayouts zeigt, wobei der Browser-Ausschnitt und der Flash-Ausschnitt gezeigt sind, und ein hervorgehobenes Wort als der diskrete Teil des zu lesenden Textes angezeigt ist, wodurch die Zentrierung des Textes innerhalb des Browser-Ausschnittes illustriert wird.

Beste Form zur Ausführung der Erfindung

[0039] Die Ausführungsform richtet sich auf ein System und ein Verfahren zum Darstellen von Text- und/oder graphischen Informationen zum Lesen unter Einsatz eines modernen Mikrocomputersystems, auf dem eine graphische Benutzeroberfläche mit einer Multitasking-Fensterbetriebsfunktion nach der Art läuft, wie sie als WINDOWS (registrierte Marke) bekannt ist.
[0040] Wie in **Fig.** 2 der Zeichnungen gezeigt, weist das System **11** im Allgemeinen ein Verarbeitungsmittel **13** in der Form eines Mikrocomputers auf, der Anzeigemittel **15** in der Form einer visuellen Anzeigeeinheit, Eingabemittel **17** in der Form einer Tastatur und einer Maus und Steuermittel **19** in der Form eines Computerprogramms hat.
[0041] Der Mikrocomputer der Verarbeitungsmittel **13** umfasst geeignete Speichermedien zum Speichern von Text- und/oder graphischen Informationen in einer elektronisch codierten Form, wie in einer Textdatei und auch zum Speichern des Computerprogramms der Steuermittel **19**.
[0042] Derartige Speichennedien können eine Harddisk, ein Band, eine CDROM, eine Floppy Disk oder dergleichen oder alternativ eine entfernte Datenbank sein, auf die mittels eines Modems oder anderer Kommunikationsschnittstellen zum Empfang und zur Verarbeitung derselben zugegriffen wird und die zur visuellen Darstellung auf der visuellen Anzeigeeinheit bereit ist.
[0043] Der Mikrocomputer kann das Computerprogramm der Steuermittel **19** in bekannter Weise laden und ein bestimmtes Textfile zum Bearbeiten und zur Darstellung der darauf gespeicherten Text- und graphischen Informationen für Lesezwecke auf der visuellen Anzeigeeinheit auswählen.
[0044] Die visuelle Anzeigeeinheit der Anzeigemittel **15** umfasst ein Anzeigemedium in der Form eines Bildschirms einer Kathodenbildröhre, eines Flüssigkristalldisplays oder Plasmadisplays, auf dem das visuelle Bild der Text- und/oder graphischen Informationen in bekannter Weise angezeigt wird.
[0045] Die Tastatur und Maus der Eingabemittel **17** werden in Kombination oder alternativ zueinander zur Eingabe von Steuersignalen für die Steuermittel **13** des Mikrocomputers zur Bedienung desselben und für das Computerprogramm der Steuermittel **19** sowie zur Änderung der Steuerparameter der Verarbeitungsmittel und Steuermittel eingesetzt.
[0046] Der Mikrocomputer der Verarbeitungsmittel **13** arbeitet unter einem geeigneten Betriebssystem wie z. B. „MS-DOS" (registrierte Marke) und einer graphischen Benutzeroberfläche mit einer Multitasking-Fensterbetriebsfunktion wie z. B. WINDOWS 3.1 (registrierte Marke), wie zuvor beschrieben. Das Computerprogramm

der Steuenrittel ist in der Form einer Anwendungssoftware, die als ein ursprüngliches WINDOWS-Programm (registrierte Marke) arbeitet, das in einer objektorientierten Programmiersprache, nämlich C++ unter Einsatz dynamisch verbundener Bibliotheken entwickelt wurde.

[0047] Demgemäß umfasst das Computerprogramm Anwendungsmittel zum Definieren des allgemeinen Layouts des Anzeigemediums für die Darstellung der Textinformationen und der visuellen Steuerattribute des Systems in Übereinstimmung mit WINDOWS 3.1 (Marke). Die Anwendungsmittel reagieren direkt auf die Steuerparameter, die für das Programm definiert sind, und rufen bestimmte Steuerfunktionen zum Definieren und Spezifizieren bestimmter Eigenschaften der Präsentation der Textinformationen innerhalb des allgemeinen Layouts auf. Diese Steuerparameter sind durch Eingabe diesbezüglicher Steuersignale mittels der Tastatur oder der Maus einstellbar und werden später in weiteren Detail beschrieben.

[0048] Das Computerprogramm umfasst auch Ausschnittmittel zum Anordnen des allgemeinen Layouts in einen oder mehreren Ausschnitten, in denen die Textinformationen dargestellt oder präsentiert werden. In dieser Hinsicht teilen die Ausschnittmittel selektiv das visuelle Layout in einen Flash-Ausschnitt und einen Browser-Ausschnitt auf, die ebenfalls später genauer beschrieben werden.

[0049] Für die Zwecke dieser Beschreibung wird die Vertrautheit mit der Computerprogrammiersprache C++, der dynamisch verlinkten Bibliothek OWL 2.0, dem Kompilieren von Programmen in Borland C++ 4.0 (Marke), dem Ablauf von WINDOWS 3.1 (Marke) und dem Ausführen von WINDOWS API Calls angenommen.

[0050] Das Computerprogramm umfasst eine Kompilierung von vier Quelldateien, nämlich ein ‚.cpp file' mit dem Hauptquellcode des Computerprogramms, einem ‚.hpp' file, das der Kopf oder Header für das Computerprogramm ist, einem ‚.rc' file, das die Resourcen des Computerprogramms darstellt und einem ‚.rh' file, das den Resourcenanfang oder-kopf des Computerprogramms darstellt. Die Kompilierung dieser vier Dateien schafft ein ‚.exe' ausführbares File und wird zusammen mit den geeigneten, dynamisch gelinkten Bibliotheken betrieben, die in Borland C++ 4.0 enthalten sind, nämlich BIDS402.DLL, OWL202.DLL, BC402RTL.DLL und CTL3DV2.DLL.

[0051] Das Programm wird mittels Menüelementen betrieben, von denen jedes äquivalente Beschleuniger oder Tastenkeys hat. Jedes Menüelement und sein äquivalenter Beschleuniger sind so aufgebaut, dass sie eine Nachricht aussenden, die eine Funktion in einer vorgeschriebenen Klasse aufruft, wie sie in einer Antworttabelle im Quellcode definiert sind.

[0052] Unten ist eine Tabelle aufgelistet, die den Menüpfad einer Auswahl in der ersten Spalte zeigt, wobei der Beschleuniger, der zu dieser Menüauswahl äquivalent ist, in der zweiten Spalte, die WINDOWS-Nachricht, die zum Aufrufen der Funktion ausgesandt wird, in der dritten Spalte und die Funktion, die durch diese Nachricht aufgerufen wird, in der vierten Spalte angegeben ist.

Menüpfad	*Kurztaste*	*Link Nachricht*	*Ruft TCharonWin Elementfunktion*
File / Open	F3	IDM_OPEN	CmFileOpen()
File / Exit	Alt+F4	24310	Standard OWL quit application
Speed / Increase	Ctrl+Up	IDM_DEC_SPEED	CmIncreaseSpeed()
Speed / Decrease	Ctrl+Down	IDM_INC_SPEED	CmDecreaseSpeed()
Speed / Update WPM	TAB	IDM_RETABULATE	CmResetWPM()
Speed / Pause or Resume	SPACE	IDM_PAUSE	CmPause()
Read / Go	ENTER	IDM_GO	CmGo()
Read / Restart	Ctrl+Home	IDM_RESTART	CmRestart()
Read / Select all		IDM_SELECTALL	CmSelectAll()
Options / Delays		IDM_DELAYS	CmDelays()
Options / Font		IDM_FONT	CmFont()
Options / Number of flashes		IDM_NUM_FLASHES	CmNumFlashes()
Options / Auto size font		IDM_AUTOSIZE	CmAutoFont()
Options / Browser window		IDM_TOPON	CmTopOn()
Help / Using help		IDM_HELPONHELP	CmHelpOnHelp()
Help / Concepts		(Not Implemented)	(Not Implemented)
Help / About		IDM_ABOUT	CmAbout()

[0053] Das Computerprogramm weist eine Anzahl von Klassen auf, die von den entsprechenden Stammklassen abgeleitet sind, die in der Borland C++ 4.0 Standardbibliothek oder in der OWL-Bibliothek vorgesehen sind. Wie in **Fig.** 2 der Zeichnungen gezeigt, gibt es sieben derartige Klassen 21a bis 21g, die ergänzend zu einem Satz von Nicht-Klassen-Funktionen 23 vorhanden sind, die allesamt zusammengruppiert worden sind.
[0054] Die sieben Klassen sind die Klasse TCharonApp 21a, die von der öffentlichen oder globalen Klasse TApplication abgeleitet ist, die Klasse TCharonWin 21b, die von der öffentlichen oder globalen Klasse TLayoutWindow abgeleitet ist, die Klasse TDelaysDialog 21c, die abgeleitet ist von der öffentlichen oder globalen Klasse TDialog, die Klasse TFlashesDialog 21d, die auch von der öffentlichen oder globalen Klasse TDialog abgeleitet ist, die Klasse TBrowserPane 21e, die abgeleitet ist von der öffentlichen oder globalen Klasse TEditFile, die Klasse TFlashPane 21f, die von der öffentlichen oder globalen Klasse TWindow abgeleitet ist, und die Klasse TString 21g, die von der öffentlichen oder globalen Klasse String abgeleitet ist.
[0055] Jede dieser Klassen wird im weiteren Detail unter Bezug auf ihre Elementfunktionen beschrieben und insbesondere auf jene, die speziell für die bestimmte Klasse abgeleitet worden sind, als auch jene, die überschriebene Funktionen sind, die von der ursprünglichen Stammklasse erhalten wurden.
[0056] Um zuerst die Klasse TCharonApp 21a zu behandeln, so wird diese Klasse von der Stammklasse TApplication abgeleitet und umfasst im Wesentlichen Anwendungsmittel zur Erzeugung der vorliegenden Computerprogrammanwendung und eine TCharonWin Instanz. Die Elementfunktionen dieser Klasse sind wie folgt:

TCharonApp	*TApplication*
TCharonApp	
InitMainWindow	~~InitMainWindow~~

1 statusBar (Public Variable)

```
TStatusBar *statusBar;
```

[0057] Dies ist der Pointer zu der Instanz der Statuszeile unten auf dem Anwendungsfenster.

2 TCharonApp (Public Constructor)

```
TCharonApp();
```

[0058] Dies baut die Computerprogrammanwendung auf.

3 InitMainWindow (Private Method)

```
void InitMainWindow();
```

[0059] Diese Funktion überschreibt die InitMainWindow() der Stammklasse. Sie ermöglicht die Anwendung von MS Windows' Ct13D DLL, die Dialogboxen in 3D anzeigt.
[0060] Sie erzeugt eine zurückgesetzte oder ausgenommene Statuszeile und weist dem Drucker die variable statusBar zu. Als nächstes erzeugt sie eine Steuerzeile durch Ausprägung eines TControlBar Objekts. „Smart Icons" werden dann in diese Zeile eingefügt. Jedes Icon, das von den Resourcen genommen wird, erzeugt eine Nachricht wenn betätigt, aber jede dieser Nachrichten hat eine Äquivalenz im Menü. Diese sind:

Icon Beschreibung	***Menüpfad***
Open a file	File/Open
Green start lights	Read/Go
Up arrow	Speed/Increase
Down Arrow	Speed/Decrease
Double vertical lines	Speed/Pause or Resume

[0061] Die Funktion geht weiter dahin, den Rahmen des Hauptanwendungsfensters durch Setzen seiner Fenstergröße von der Computerprogramminitialisierungsdatei zu binden und ihm ein Menü, Icon und Beschleuniger bzw. Kurztasten von den Resourcen zuzuweisen.
[0062] Beides, die Breite und die Höhe des Hauptfensters werden aus den Einträgen Width und Height in der Initialisierungsdatei entnommen (siehe TCharonWin::CanClose()). Der Eintrag Hint wird als Parameter zum Aufruf SetHintMode() verwendet.
[0063] Die nächste Klasse TCharonWin 21b ist, wie zuvor beschrieben, von der TLayoutWindow Klasse abgeleitet. Sie beinhaltet zwei Anzeigeausschnitte, nämlich den oberen Ausschnitt (eine Instanz einer TBrowserPane, einem Abkömmling einer TEditFile Klasse), die ein ASCII Textfile enthält und anzeigt, und den unteren Ausschnitt (eine Instanz einer TFlashPane, einem Abkömmling von Twindow), die ein Wort oder eine Linie von Worten anzeigt.
[0064] Diese Klasse ist ein friend der TFlashPane.
[0065] Die Elementfunktionen dieser Klasse sind wie folgt:

TCharonWin	*TLayoutWindow*
TCharonWin	
~TCharonWin	
SetupWindow	~~SetupWindow~~
CanClose	~~CanClose~~
SetupPanes	
ReconfigureWindow	
IsFlashing	
SetDisplayFont	
DisplayNextLine	
DelayUntilTime	
FlashLoop	
PauseFlashing	
ShowWordsPerMinute	
WordsPerMinute	
EvSize	~~EvSize~~
CmFileOpen	
CmGo	
CmSelectAll	
CmRestart	
CmDelays	
CmFont	
CmNumFlashes	
CmAutoFont	
CmTopOn	
IncreaseAWord	
CmIncreaseRate	
CMDecreaseRate	
CmPause	
CmResetWPM	
CmHelpOnHelp	
CmAbout	

1 rates (Public Variable)

```
TDelayRates rates;
```

[0066] Das ist eine Instanz oder Ausprägung der die Verzögerungszeiten für die verschiedenen Bedingungen enthaltenden Struktur. Die Elemente der Struktur und ihr Zweck ist wie unten angegeben:

Elementvariable	*Zweck*
UINT interval	Verzögerung zwischen zwei Wortblitzen. Dieses Intervall berücksichtigt nicht die Ausblendezeit zwischen Worten.
UINT blackout	Verzögerung eines leeren Bildschirms zwischen zwei Blitzen.
UINT punctuations	Verzögerung, wenn eine Punktuation als das letzte oder erste Zeichen eines Blitzes erkannt wird.
UINT ends	Verzögerung, falls das letzte Zeichen eines Aufblitzens eine Punktuation für ein Satzende ist.
UINT cr	Verzögerung, falls die nächsten zwei Zeichen im Datenfluss Zeilenschaltungen sind.

2 fileData (Private Variable)

```
TOpenSaveDialog::TData fileData;
```

[0067] Das ist eine Instanz der Struktur, die verwendet wird, um den Dateinamen zum Lesen zu erhalten. Siehe die OWL-Klasse TFileOpenDialog.

3 fontData (Private Variable)

```
TChooseFontDialog::TData fontData;
```

[0068] Dies ist eine Instanz der Struktur, die verwendet wird, um die Font-Attribute angezeigt zu erhalten. Siehe OWL-Klasse TchooseFontDialog.

4. browserpane (Private Variable)

```
TbrowserPane browserpane;
```

[0069] Diese hält eine Instanz oder Ausprägung des oberen Anzeigeausschnitts. Siehe Klasse TBrowserPane.

5 flashpane (Private Variable)

```
TFlashPane flashpane;
```

[0070] Diese hält eine Instanz oder Ausprägung des unteren Anzeigeausschnitts. Siehe Klasse TFlashPane.

6 flashing (Private Variable)

```
FLASH_LOOP flashing;
```

[0071] FLASH_LOOP ist ein aufnummerierter Typ von FLASHING, NOT_FLASHING oder QUIT. Die variable flashing ist eine Flagge zur Anzeige, ob der untere Ausschnitt Worte aufleuchten oder aufblitzen lässt oder nicht. Die Kenntnis dieses Status wird an verschiedenen Stellen innerhalb des Programms benötigt, so z. B. wenn der Benutzer seinen Cursor in den oberen Ausschnitt zurückstellt. Der QUIT-Status wird gesetzt, wenn der Benutzer wählt, die Anwendung zu verlassen.

7 disFont (Private Variable)

```
Tfont *dispFont;
```

[0072] Diese ist der Zeiger für den Font, der für die Anzeige eines Flashs im unteren Ausschnitt benutzt wird.

8 autoSize (Private Variable)

```
BOOL autoSize;
```

[0073] Dies ist eine Boolesche Flagge für das Einstellen der Benutzerfontgröße, wie sie im unteren Ausschnitt verwendet wird. Ein TRUE zeigt an, dass der Font hinsichtlich seiner Größe im Verhältnis zum Fenster gesetzt ist, und ein FALSE zeigt an, dass der Font gemäß der Größenspezifikation des Benutzers in fontData gezeichnet wird.

9 topOn (Private Variable)

```
BOOL topOn;
```

[0074] Dies ist eine Boolesche Flagge einer Benutzereinstellung des oberen Ausschnitts. Ein TRUE zeigt an, dass der obere Ausschnitt angezeigt werden soll, und ein FALSE zeigt an, dass der obere Ausschnitt nicht gezeigt werden soll.

10 hasTopPane (Private Variable)

```
BOOL hasTopPane;
```

[0075] Dies ist eine Boolesche Flagge, die den Anzeigestatus des oberen Ausschnitts zeigt. Ein TRUE zeigt an, dass der obere Ausschnitt derzeit angezeigt wird (unabhängig davon, ob topPane TRUE oder FALSE ist) und ein FALSE zeigt an, dass der obere Ausschnitt derzeit ausgestaltet ist. Der Unterschied zwischen dieser und der topOn Flagge ist, dass diese Flagge den derzeitigen Status der Anwendung anzeigt, und topOn anzeigt, was der Benutzer bevorzugt. In Folge dessen wird, wenn die Anwendung in den Pausemodus geht, der obere Ausschnitt unabhängig von der vom Benutzer bevorzugten Einstellung angezeigt.

11 ifs (Private Variable)

```
ifstream *ifs;
```

[0076] Dies ist ein Zeiger zur Handhabung des zu lesenden Files.

12 strln (Private Variable)

```
Tstring strIn;
```

[0077] Diese ist eine Ausprägung von Tstring. Diese hält das derzeitige Wort oder die derzeitigen Worte, die aufgeblitzt oder hervorgehoben werden. Siehe Tstring.

13 pauseTick (Private Variable)

```
DWORD pauseTick;
```

[0078] Dies umfasst das Setzen des Akkumulationstimers, während die Anwendung im Pausenmodus ist. Dies wird getan, um eine genaue Worte-Pro-Minute (WPM) Zählung zu erhalten.

14 initialTick (Private Variable)

```
DWORD initialTick;
```

[0079] Dies umfasst das Starten des Tickens des Anfangszählers, der die Worte-Pro-Minute (WPM) Zählung startet.

15 wordCount (Private Variable)

```
UINT wordCount;
```

[0080] Diese hält die Anzahl der in dem unteren Ausschnitt aufgeleuchteten Worte seit dem Zurücksetzen des initialTick.

16 pauseStarted (Private Variable)

```
BOOL pauseStarted;
```

[0081] Dies ist eine Boolesche Variable, die anzeigt, ob das Flash-Lesen durch ein „unpause" (Pause aufheben) im Gegensatz zu einem „go" (Starten) gestartet wurde. Ein TRUE zeigt an, dass das Aufleuchten in CmPause() gestartet wurde und ein FALSE zeigt an, dass das Aufleuchten in CmGo() gestartet wurde. Dies wird gemacht, da ein „unpause" und ein „go" zwei verschiedene Merkmale sind und zwei verschiedene Verhalten haben und weil die Anwendung das letzte Ereignis kennen muss, das sie aktiviert. Falls das Aufleuchten durch ein unpause in CmPause() aktiviert wird, fährt es fort, Worte aufleuchten oder aufblitzen zu lassen hinter seinen Endanwendungsmarkierungen, während ein „go" das Aufleuchten stoppen würde, wenn es die Endanwendungsmarkierung erreicht.

17 delay_ms (Private Variable)

```
UINT delay_ms;
```

[0082] Diese hält die zu verwendende Anzahl als Parameter für Delay() (Verzögerung) fest, so dass ein Aufruf an Delay() mit diesem Wert eine Millisekunde ist (siehe Delay()). Diese Variable wird berechnet und nur einmal

im SetupWindow() initiiert, wenn die Anwendung startet bzw. hochfährt. Wie in der Beschreibung in Delay() erklärt, wird diese lokale Schleifenverzögerung verwendet, um die 55 Millisekundenschranke zu umgehen.

18 TCharonWin (Public Constructor)

```
~TCharonWin();
```

[0083] Dies löscht die Instanzen oder Ausprägungen von ifstream (*if, die Datei zum Anzeigen) und Tfont (*dispFont, den in der Anzeige im unteren Ausschnitt verwendeten Font).

19 ~TCharonWin (Private Destructor)

```
~TCharonWin();
```

[0084] Dies löscht die Instanz oder Ausprägung von ifstream (*if, die Datei zum Anzeigen) und Tfont (*dispFont, den in der Anzeige im unteren Ausschnitt verwendeten Font).

20 SetupWindow (Private Method)

```
void SetupWindow();
```

[0085] Diese Funktion überschreibt die Stammfunktion SetupWindow(). Nach Aufruf der zugrunde liegenden Funktion erneuert es die Statuszeile und checkt und überprüft dann zwei Menüeigenschaften. Die manipulierten Menüeigenschaften sind wie unten angegeben:

Menüpfad	***Geprüft auf Bedingung ...***
Options/Auto size font	autoSize ist TRUE
Options/Browser window	topOn ist TRUE

[0086] Am Ende führt sie einen Zeittest zur Berechnung des Werts von delay_ms durch. Um dies zu tun, bekommt sie die derzeitige Einstellung (tick) vom System und ruft Delay() mit einer großen Zahl als Parameter auf. Wenn sie zurückkommt, bekommt sie die derzeitige Einstellung wiederum vom System. Der Unterschied zwischen dieser und der vorherigen Systemeinstellung ist die Anzahl von Zeiteinheiten, die verstreichen, während Delay() läuft. Die ursprüngliche, als Parameter an Delay() gegebene Zahl geteilt durch die Tickdifferenz ist der Wert, der an Delay() gegeben werden muss, um es um eine Millisekunde zu verzögern. Graphisch in Pseudo-Code bedeutet dies:

```
origTick = Get original clock tick from system


Call Delay() with arbitrary large number n


delay_ms = n / (clock tick now - origTick)
```

21 CanClose (Private Method)

```
BOOL CanClose();
```

[0087] Diese Funktion überschreibt die Stammfunktion. Zuerst ruft sie das zugrunde liegende CanClose() auf und, wenn das Hauptfenster aus irgendeinem Grund nicht geschlossen werden kann, springt diese Funktion zurück.

[0088] Einige Werte werden im Dateiinitialisierungsdatensatz, CHARON.INI. gesichert. Unter diesen sind die Breite und die Höhe des Fensters. Die Fensterkoordinaten am Ende dieser Sitzung können jedoch nicht die aktuellen Koordinaten zu Beginn der nächsten Sitzung sein. Der Grund dafür ist, dass während eines Pausemodus der obere Ausschnitt aufspringt (wie ein Pop-up-Menü), selbst wenn die Browserfensteroption ausgeschaltet ist. Eine Überprüfung ist nötig, damit die Benutzerflagge (topOn) konsistent ist mit dem derzeitigen Fensterstatus (hasTopPane). Wenn die Benutzereinstellung aus ist und es derzeit einen oberen Ausschnitt gibt, wird ein Aufruf an ReconfigureWindow() gemacht, um das Hauptfenster in seiner Größe zu ändern, bevor

die Koordinaten genommen werden.
[0089] Die Tabelle unten zeigt die in CHARON.INI. gespeicherten Variablen:

Variablenname in Code	*Eintragsname in CHARON.INI*	*Wo verwendet*
autoSize	Autosize	TCharonWin::TCharonWin()
blackout	Blackout	TCharonWin::TCharonWin()
punctuations	Punctuations	TCharonWin::TCharonWin()
ends	Sentence End	TCharonWin::TCharonWin()
cr	Paragraph End	TCharonWin::TCharonWin()
interval	Interval	TCharonWin::TCharonWin()
strIn.numFlashes	Flash	TCharonWin::TCharonWin()
fontData.LogFont.lfHeight	Font Height	TCharonWin::TCharonWin()
fontData.LogFont.lfFaceName	Font Face	TCharonWin::TCharonWin()
Attr.W	Width	TCharonApp::InitMainWindow()
Attr.H	Height	TCharonApp::InitMainWindow()

[0090] Diese Funktion fordert dann WINHELP zur Beendigung auf.
[0091] Unter normalen Umständen verursacht ein wiederkehrendes TRUE in CanClose(), die Anwendung zu beendigen. In dieser Anwendung kann jedoch CanClose() aus der Funktion FlashLoop() aktiviert werden. Dazu wird IsFlashing() aufgerufen, um zu wissen, ob dies der Fall war. Wenn ja, wird eine Flagge (Setzen flashing zu QUIT) gesetzt und CanClose() springt zurück zu FALSE um FlashLoop() zu informieren, die Anwendung zu beenden (FlashLoop() ruft die API PostQuitMessage(), um dies zu veranlassen und umgeht somit jeden weiteren Aufruf von CanClose()). Wenn das Aufleuchten oder Aufblitzen nicht aktiv ist, macht CanClose() einen normalen Ausgang.
[0092] Als Anmerkung, CanClose() kann nicht zu TRUE zurückspringen, wenn es ein Signal zum Schließen erhalten hat, während ein Aufleuchten oder Aufblitzen im Gange ist. Dies verursacht einen Schutzfehler.

22 SetupPanes (Private Method)

```
void SetupPanes(BOOL expand);
```

[0093] Diese Funktion beeinflusst die Größenverhältnisse der oberen und unteren Ausschnitte. Der Parameter expand bestimmt, ob der obere Ausschnitt als 65%-Anteil gezeigt werden soll oder ob er nicht gezeigt werden soll, indem ihm ein 0%-Verhältnis gegeben wird. Siehe OWL's TLayoutWindow im Hinblick auf das Festsetzen von Verhältnisangaben.
[0094] Beachte, dass diese Funktion der Layoutmetrik Werte zuweist und die Stammfunktion SetChildLayoutMetric() aufruft, um diese Werte den beiden Ausschnitten zuzuordnen. Sie ruft nicht die Stammfunktion Layout() auf, um die Ändenangen erneut anzuzeigen.

23 ReconfigureWindow (Private Method)

```
void ReconfigureWindow(BOOL expand);
```

[0095] Diese Funktion wird aufgerufen, um das Hauptanwendungsfenster auszudehnen und zusammenzuziehen. Falls der Parameter expand TRUE ist, wird der obere Ausschnitt mit einem 65%-Verhältnis des Hauptfensters angezeigt, während der untere Ausschnitt in derselben absoluten Größe verbleibt. Das Hauptfenster wird daher vergrößert. Falls der Parameter expand FALSE ist, wird der obere Ausschnitt mit einem 0%-Verhältnis des Hauptfensters angezeigt, während der untere Ausschnitt in derselben absoluten Größe verbleibt. Das Hauptfenster wird daher verkleinert.
[0096] Dies erfolgt dadurch, dass man die Fenstergröße des Mutterfensters (d. h. Hauptfensters) und die Größe des unteren Ausschnitts nimmt. Dieses Mutterfenster wird dann durch den Aufruf MoveWindow() mit den derzeitigen linken und oberen Koordinaten der Mutter und der derzeitigen Breite hinsichtlich der Größe neu bestimmt. Die Höhe ist die derzeitige Höhe des Ausgangsfensters mit dem Anteil der Höhe des unteren Ausschnitts wie in folgender Berechnung:

```
Höhe des Ausgangsfensters +/- Höhe des unteren Ausschnitts *

65% / (100-65%)
```

[0097] Der Anteil entweder positiv sein, wenn das Fenster vergrößert werden soll oder negativ, wenn das

Fenster reduziert werden soll. Die obige Berechnung betrifft nur die Größenanpassung Hauptanwendungsfensters. Die Größenanpassung der inneren zwei Ausschnitte erfolgt durch Aufruf von SetupPanes() und TLayoutWindow::Layout().

24 IsFlaching(Private Method)

```
BOOL IsFlashing();
```

[0098] Wird zu TRUE, falls der untere Ausschnitt aufleuchtender oder aufblitzender Text ist (d. h. die Variable flashing hat den Status FLASHING).

25 SetDisplayFont (Private Method)

```
void SetDisplayFont(int height);
```

[0099] Diese Funktion wird durch TFlashPane::Paint() aufgerufen und erzeugt den Font, der zur Anzeige von Text im unteren Ausschnitt verwendet wird. Dies ist nur so, wenn die Variable dispFont NULL ist, ansonsten endet sie. Falls die Benutzeroption , autoSize' gesetzt ist, wird die TFont-Ausprägung oder -Instanz mit dem Parameter height erzeugt. Ansonsten wird dieser Parameter ignoriert und die Ausprägung mit den Fontmaßen in fontData erzeugt.

26 DisplayNextLine (Private Method)

```
int DisplayNextLine();
```

[0100] Diese Funktion bekommt und zeigt das nächste Wort oder die nächsten Worte (wie festgelegt durch die Variable wordCount) an. Sie meldet die Zeit in Millisekunden, die diese Textzeile angezeigt werden muss. Beide Ausschnitte geben den Wechsel in der Anzeige wieder.
[0101] Die Funktion GetAFlash() der Instanz oder Ausprägung strln wird aufgerufen, um eine neue Textzeile zu erhalten. Bevor sie mit der Anzeige fortfährt, überprüft sie, ob das Aufleuchten oder Aufblitzen durch ein „pause" oder „go" gestartet wurde. Falls es durch ein „go" gestartet wurde, vergleicht die Funktion die derzeitige Dateiposition mit dem Ende des Benutzerauswahlblocks (siehe BlockEnd() der Instanz oder Ausprägung manyWordsPane), um zu verifizieren, ob die Textzeile an oder jenseits der Benutzerendauswahlkennzeichnung ist. Falls ja, erfolgt keine Anzeige.
[0102] Auf jeden Fall, falls die Funktion nicht in der Lage ist, mehr Worte zur Anzeige zu erhalten, ruft sie PauseFlashing() auf, um das Hervorheben, Aufleuchten oder Aufblitzen zu stoppen, und ruft SelectBlock() der Ausprägung manyWordsPane, um den Block zu kennzeichnen, der die ursprüngliche Auswahl der Benutzers im oberen Ausschnitt war.
[0103] Wenn eine Anzeige möglich ist, wird der Text in beiden Ausschnitten wieder gegeben. Für den oberen Ausschnitt kümmert sich hierum der Aufruf PresentLine() der Ausprägung manyWordsPane. Für den unteren Ausschnitt kümmert sich hierum der Aufruf TFlashPane::RepaintWindow(). Die Gesamtzahl der Blitze (wordCount) wird um die Anzahl der Blitze vermehrt, die von GetAFlash() gelesen werden.
[0104] Bevor diese Funktion endet, wird eine Berechnung des Betrags an Verzögerung durchgeführt, die erfolgen muss, bevor eine neue Textzeile angezeigt werden kann.
[0105] Diese Verzögerung ist das interval plus eine Extraverzögerung. Wenn die Textzeile ein Satzende ist (siehe IsSentenceEnd()) und wenn das nächste Zeichen im Zeichenfluss eine neue Zeile ist (d. h. eine Zeilenschaltung), ist die Zeile auch das Ende eines Absatz. Damit ist die Extraverzögerung diejenige eines Absatzendes (rates.cr) oder eines Satzendes (rates.ends), je nachdem welche von beiden größer ist. Falls jedoch das erste oder letzte Zeichen der Textzeile eine Punktuation ist (siehe IsPunctuation()), dann ist die Extraverzögerung die einer normalen Punktuation (rates.punctuation).

27 DelayUntilTime (Private Method)

```
void DelayUntilTime(UINT xTime);
```

[0106] Diese Funktion unterbricht das Programm intelligent für einen gegebenen Zeitbetrag (in Millisekunden) durch Aufsetzen einer lokalen Verzögerungsschleife.
[0107] Falls der Parameter xTime größer als 55 ms ist, d. h. mehr als ein Taktzyklus in Windows, ruft sie kontinuierlich die API GetTickCount() auf, bis die Verzögerungszeit erreicht ist, oder, wenn weniger als 55 ms verblieben sind, wird die TApplication::PumpWaitingMessages() zwischen Zeitüberprüfungen aufgerufen, um an-

dere Anwendungen einzubringen (und somit ein Einfrieren zu verhindern), während diese Funktion wartet.
[0108] Falls xTime oder die verbleibende Verzögerungszeit weniger als 55 ms ist, wird die Elementfunktion Delay() aufgerufen, um diesen Betrag durch einen Maschinenauftragschleifenprozess (siehe Delay()) zusammenzuschrumpfen.

28 FlashLoop (Private Method)

```
BOOL FlashLoop();
```

[0109] Diese Funktion ist der Kern der Anwendung und wird durch CmGo() und CmPause() aufgerufen. Im Wesentlichen erhält sie von der Datei eine Textzeile, zeigt sie an, blendet sie aus und überprüft, dass sie sich noch nicht in einer End- oder Exitbedingung befindet.
[0110] Zuerst wird CmResetWPM() aufgerufen, um den WPM-Zähler zu initialisieren. Dann gelangt sie in eine Schleife, die sicher stellt, dass die Anwendung in einem Flashing-Modus oder Blitzmodus ist (durch Aufruf IsFlashing()). Während sie im Flashing-Modus ist, ruft sie SelectionChanged() der Instanz oder Ausprägung manyWordsPane auf, um zu sehen, ob die Auswahlmarkierungen im oberen Ausschnitt dieselben sind wie sie zuvor der Fall waren. Falls nicht, hat der Benutzer den Cursor oder den Auswahlblock neu angeordnet. Wenn dies geschieht, bekommt sie die Startmarkierung und ordnet den Dateizeiger (mit ifstream:seekg()) der Textposition neu zu, die denselben Offset wie die Markierung hat.
[0111] Sie ruft dann DisplayNextLine() auf, um eine Textzeile zu lesen und stellt sie in beiden Ausschnitten dar. DisplayNextLine() meldet die Verzögerung, die eingehalten werden muss, bevor die Ausschnitte wieder upgedatet werden können. Diese Meldung wird der Parameter, um DelayUntilTime() aufzurufen, der für die vorgegebene Zeit schläft.
[0112] Wenn die entsprechende Verzögerung erfolgt ist, wird der untere Ausschnitt ausgeblendet. Dies erfolgt dadurch, dass zuerst strln gelöscht wird und dann TFlashPane::RepaintWindow() für den unteren Ausschnitt aufgerufen wird, was TFlashPane::Paint() updatet und letztlich aufruft. DelayUntilTime() wird aufgerufen, um die Einstellzeit für das Ausblenden des unteren Ausschnitts rates.blackout zu verzögern.
[0113] Als nächstes wird die Statuszeile mit einem Aufruf ShowWordsPerMinute() upgedatet.
[0114] Im Fall von sehr kleinen Verzögerungswerten wie z. B. durchgängigen Nullverzögerungen, zeigt die Anwendung kontinuierlich an, ohne auf irgendwelche Windowsnachrichten zu reagieren. Damit würden alle anderen MS Windows-Anwendungen einfrieren einschließlich ihr selbst. Um dies zu verhindern, wird die TApplication::PumpWaitingMessages() aufgerufen. Dies erfolgt alle 15 Schleifen, so dass der WPM-Zähler nicht zu schief ist, da PumpWaitingMessages() etwas Zeit zur Verarbeitung benötigen kann.
[0115] Zuletzt vor einer nochmaligen Schleife überprüft diese Funktion, ob die Anwendung beenden will (Variable flashing hat den Wert QUIT). Falls dies wahr ist, sendet sie eine Endnachricht mit dem API-Ruf PostQuitMessage() und unterbricht das Aufleuchten, so dass diese Schleife beendet werden kann.

29 PauseFlashing (Private Method)

```
void PauseFlashing();
```

[0116] Diese Funktion setzt die Flagge, um das Aufleuchten oder Aufblitzen von Text zu stoppen, und setzt beide Auswahlmarkierungen im oberen Ausschnitt auf die derzeitige Position wie im Dateistrom (d. h. Aufheben Blockauswahl). Diese Funktion arbeitet nur, wenn das Aufleuchten derzeit aus ist.

30 ShowWordsPerMinute(Private Method)

```
void ShowWordsPerMinute();
```

[0117] Diese Funktion ruft die Elementfunktion WordsPerMinute() auf, um den derzeitigen WPM-Zähler zu erhalten, setzt ihn auf in einen C-String und ruft TStatusBar::SetText() auf, um ihn in die Statuszeile zu schreiben.

31 WordsPerMinute(Private Method)

```
UINT WordsPerMinute();
```

[0118] Diese Funktion meldet den derzeitigen WPM-Zählerstand. Der Wortzähler je Millisekunde wird festgestellt, indem man die Differenz zwischen dem derzeitigen Zählerstand oder Tickerstand (durch Aufrufen der API-Funktion GetTickCount()) und dem Zählerstand zu Beginn der Zählerstatistik (initialTick) bildet und dann invers durch die Anzahl der während dieser Periode gelesenen Worte teilt. Diese Zahl wird dann mit 60.000

multipliziert, um sie in WPM umzuwandeln. Als Vorsichtsmaßnahme gegen einen „Teile durch Null"-Fehler wird der derzeitige Stand verglichen gegenüber dem anfänglichen Stand, und falls sie derselbe sind, führt sie keine Berechnung durch, sondern kehrt zu einem Nullstand des WPM zurück.

32 EvSize (Private Method)

```
void EvSize(UINT sizeType, TSize& size);
```

[0119] Diese Funktion überschreibt die Stammfunktion. Sie sichert, dass der angezeigte Font (dispFont) gelöscht wird auf jede WM SIZE-Nachricht (d. h. auf eine Größenänderung des Hauptfensters durch den Benutzer) hin, indem sie den Font löscht und zu Null setzt, so dass er in der Funktion Paint() der Instanz oder Ausprägung flashpane, und zuletzt in SetDisplayFont() später neu erzeugt wird. Sie ruft dann die Basisklasse EvSize() auf.

33 CmFileOpen (Private Method)

```
void CmFileOpen();
```

[0120] Diese Funktion wird auf einen Dateiöffnungsauftrag aufgerufen. Sie erzeugt und führt eine Instanz des TFileOpenDialog() auf, indem sie die TData-Instanz fileData verwendet. Falls sie IDOK zurückgibt, wird der alte Datenfluss gelöscht, sofern einer vorhanden ist, und der Aufleuchtestatus wird gestoppt mit PauseFlashing(). Ein neuer Dateistrom wird durch den Aufruf des Constructors ifstream mit dem Parameter FileName erzeugt, der von TFileOpenDialog() erhalten wird. Falls der Aufbau erfolgreich ist, wird der obere Ausschnitt aufgefordert, dieselbe Datei zu lesen. Dies erfolgt durch Aufrufen von SetFileName() und Read() der Instanz oder Ausprägung browserpane.
[0121] Als nächstes wird der Dateiname an den Fenstertitel der Anwendung angehängt mit dem API-Aufruf SetWindowText(). Dann wird CmSelectAll() aufgerufen, um sämtlichen Text im oberen Ausschnitt zu blockieren und die horizontale Scroll-Leiste zu entfernen.
[0122] Falls ifstream nicht instantüert werden kann, d. h. die Datei nicht geöffnet werden kann, springt ein Fehlerdialog auf (unter Einsatz der API MessageBox()). Ein derartiger Fall sollte nie geschehen, da TFileOpenDialog() die Dateizugänglichkeit checkt, bevor ifstream sie öffnet.
[0123] Jedoch kann browserpane fehlschlagen, wenn sie versucht Read() aufzurufen, um die ganze Datei in den oberen Ausschnitt einzulesen. Dies geschieht, falls die zu lesende Datei 64K überschreitet, da der obere Ausschnitt eine Instanz eines standardmäßigen OWL Klassenobjekts TEditFile() ist (siehe Klasse TFlashPane).

34 CmGo (Private Method)

```
void CmGo();
```

[0124] Diese Funktion wird aufgerufen, um das Lesen zu starten oder neu zu starten. Es fährt nur dann fort, wenn es eine geöffnete Datei gibt (d. h. ifstream nicht NULL ist) und falls das Lesen nicht bereits aktiv ist (d. h. IsFlashing() FALSE meldet). Es gibt Umstände, unter denen die Benutzerflagge für den oberen Ausschnitt ausgeschaltet ist (d. h. topOn ist FALSE), jedoch der Ausschnitt tatsächlich gezeigt wird (d. h. hasTopPane ist TRUE) wie z. B. während einer Pause. Falls dies geschehen sollte, wird der obere Ausschnitt durch den Aufruf ReconfigureWindow() verborgen.
[0125] Wenn dies erfolgt, werden Aufrufe an Funktionen gemacht, um festzuhalten, ob die Blockauswahl des oberen Ausschnitts sich ändert, um das Aufleuchten zum erneuten Starten an den Anfang des Blocks zu setzen und um die WPM-Zählervariable zurückzusetzen (siehe TBrowsePane::BlockChanged(), CmRestart() und CmResetWPM()).
[0126] Schließlich setzt sie eine Flagge zur Anzeige, dass das Aufleuchten oder Flashing nicht gestartet wurde mit einer Pauseaufhebtaste (d. h. pauseStart ist FALSE) und ruft dann die Kernflashschleife FlashLoop().

35 CmSelectAll (Private Method)

```
void CmSelectAll();
```

[0127] Diese Funktion wird auf den Befehl hin aufgerufen, den gesamten Text im oberen Ausschnitt auszuwählen. Sie ruft einfach SelectAll() der browserpane-Instanz auf.

36 CmRestart (Private Method)

```
void CmRestart();
```

[0128] Diese Funktion wird auf den Befehl hin aufgerufen, das Aufleuchten neu zu starten. Falls es einen Dateifluss gibt, wird das Aufleuchten gestoppt (mit PauseFlashing()). Der Dateistrom wird neu positioniert auf die Startanwendungsmarkierung des oberen Ausschnitts. Beide API-Markierungen für den Anfang und das Ende werden auf die Stellung der Anfangsanwendungmarkierung gesetzt (mit Aufruf von SetSelection()), so dass der API-Block verschwindet und der ‚I'-Strahl (Cursor) auch an diese Position bewegt wird.

37 CmDelays (Private Method)

```
void CmDelays();
```

[0129] Diese Funktion wird auf den Befehl hin aufgerufen, den Benutzer diese Verzögerungsraten editieren zu lassen. Ein Dialog von Verzögerungsraten (siehe Klasse TDelaysDialog) wird erzeugt und ausgeführt. Falls OK gedrückt wird, werden die Variablen für die WPM-Berechnung zurückgesetzt (siehe CmResetWPM()).

38 CmNumFlashes (Private Method)

```
void CmNumFlashes();
```

[0130] Diese Funktion wird auf den Befehl hin aufgerufen, den Benutzer die Anzahl der Worte, die auf dem unteren Ausschnitt gesehen werden, zu editieren. Ein Dialog TFlashesDialog wird erzeugt und ausgeführt, der den Parameter der derzeitigen Anzahl von Blitzen oder Aufleuchten angibt. Falls OK gedrückt wird, erhält strin.numFlashes den neuen Wert.

39 CmFont (Private Method)

```
void CmFont();
```

[0131] Diese Funktion wird auf den Befehl hin aufgerufen, den Benutzer den Font auswählen zu lassen, der für die Anzeige im unteren Ausschnitt verwendet wird. Ein allgemeiner Dialog der erhältlichen Fonts (siehe Klasse TDelaysDialog) wird erzeugt und ausgeführt. Falls OK gedrückt wird, wird der derzeitige Font gelöscht und zu Null gesetzt (siehe dispFont). Damit würde TFlashPane::Paint() veranlasst einen Font mit der vom User ausgewählten Fontmetrik in fontData neu zu erzeugen.

40 CmAutoFont (Private Method)

```
void CmAutoFont();
```

[0132] Diese Funktion wird auf die Bitte hin aufgerufen, den Benutzer die Auto-Font-Größenänderung wählen oder abwählen zu lassen. Sie ist eine Umschaltfunktion, die die Variable autoSize zu TRUE oder FALSE bei wechselnden Aufrufen setzt. Das Menüelement wird ebenfalls angekreuzt oder nicht angekreuzt mit CheckMenuItem(), wenn diese Variable hin- und hergeschaltet wird.
[0133] Bei jedem Aufruf wird der derzeitige Anzeigefont dispFont gelöscht und Null gesetzt, so dass TFlashPane::Paint() den Anzeigefont in der korrekten Größe neu erzeugen kann.

41 CmTopOn (Private Method)

```
void CmTopOn();
```

[0134] Diese Funktion wird auf die Bitte hin aufgerufen, den Benutzer die Anzeige des oberen Ausschnitts an- und ausschalten zu lassen. Wenn diese Funktion aufgerufen wird, um den oberen Ausschnitt anzuschalten und falls der obere Ausschnitt nicht bereits gezeigt wird (wie z. B. während einer Pause), ruft sie ReconfigureWindow() mit einem TRUE Parameter auf, um den oberen Ausschnitt anzuzeigen. Ebenso falls diese Funktion aufgerufen wird, um den unteren Ausschnitt auszuschalten und falls der obere Ausschnitt nicht bereits ausgeschaltet ist, ruft sie ReconfigureWindow() mit einem FALSE Parameter auf, um den unteren Ausschnitt zu „entfernen".
[0135] Der Wert der Variablen topOn wird umgeschaltet und das Menüelement wird angestrichen oder nicht

angestrichen mit CheckMenuItem().

42 IncreaseAWord (Private Method)

```
void IncreaseAWord(SIGN sign)
```

[0136] Diese Funktion berechnet das Anzeigezeitintervall (in Millisekunden), das zwischen zwei Blitzen oder zweimaligem Aufleuchten (siehe rates.interval) benötigt wird, so dass der Wechsel in der WPM-Rate etwa 1 WPM ist. Beachte, dass, wenn das Anzeigezeitintervall bereits sehr kurz ist, eine Variation von einer Millisekunde eine Veränderung in der WPM-Rate von mehr als 1 WPM verursachen kann. In diesem Fall ist das Intervall eine voreingestellte Millisekunde, MIN INTERVAL.
[0137] Der Parameter sign ist entweder NEGATIV oder POSITIVE, was im Wesentlichen die Variation bestimmt, um ein Intervall zu sein, das größer oder kleiner wird.
[0138] Die Funktion berechnet zuerst die derzeitige WPM-Zählung und, falls sie nicht Null ist (um einen Verarbeitungsfehler „Teile durch Null" auszuschließen), wird rates.interval überprüft, ob sie innerhalb eines Bereichs ist, wie er durch MIN_INTERVAL und MAX_INTERVAL spezifiziert ist. Falls sie sich nicht in diesem Bereich befindet, wird sie berichtigt, so dass dies wahr ist. Dies ist so, um das Intervall innerhalb eines erkennbaren Bereichs zu halten. D. h., dass es außerhalb dieses Bereich keine wahrnehmbare Differenz in der Geschwindigkeit für den Benutzer der Anwendung gibt.
[0139] Eine Berechnung wird durchgeführt durch Teilen des derzeitigen (berichtigten) interval durch die derzeitige WPM, um herauszufinden, wie viel interval vergrößert oder verringert werden muss. Falls diese Änderung (delta) zu klein ist (d. h. kleiner als Null), ist das Intervall einfach um eins zu vergrößern oder zu verkleinern.
[0140] Bevor diese Veränderung zum derzeitigen interval hinzugefügt wird, überprüft sie, dass das neue Intervall nicht weniger als das MIN_INTERVAL ist, und besonders von Bedeutung, dass das neue Intervall nicht umlaufend ist (da interval eine unbezeichnete Integer ist). Falls sie weniger als MIN INTERVAL ist, wird interval auf diesen Wert gesetzt.
[0141] Schließlich wird CmResetWPM() aufgerufen, um die Variablen für eine neue WPM-Zählung zurückzusetzen.

43 CmIncreaseRate (Private Method)

```
void CmIncreaseRate();
```

[0142] Diese Inlinefunktion wird auf die Aufforderung aufgerufen, den Benutzer die Aufleuchtgeschwindigkeit erhöhen zu lassen. Sie ruft einfach IncreaseAWord() mit dem Parameter von POSITIVE auf, um ein positives Delta anzuzeigen.

44 CmDecreaseRate (Private Method)

```
void CmDecreaseRate();
```

[0143] Diese Inlinefuktion wird auf die Bitte hin aufgerufen, den Benutzer die Aufleuchtgeschwindigkeit verringern zu lassen. Sie ruft einfach IncreaseAWord() mit dem Parameter NEGATIVE auf, um ein negatives Delta anzuzeigen.

45 CmPause (Private Method)

```
void CmPause();
```

[0144] Diese Funktion wird auf die Bitte hin aufgerufen, das Aufleuchten oder Aufblitzen zu stoppen oder zu starten. Falls es keinen Dateistrom gibt und IsFlashing() TRUE meldet, wird das Aufleuchten pausiert. Ansonsten falls es einen Dateistrom gibt und IsFlashing() FALSE meldet, wird das Aufleuchten neu gestartet oder die Pause beendet.
[0145] Falls im Falle eines Pausierens derzeit kein oberer Abschnitt vorhanden ist (d. h. hasTopPane FALSE ist), wird der obere Ausschnitt mittels ReconfigureWindow() eingeschaltet. TBrowserPane::Scroll() wird aufgenafen, um den oberen Ausschnitt mehrere Male nach links zu scrollen, weil dadurch sicher gestellt wird, dass Spalte eins sichtbar ist, falls sie es nicht ohnehin schon ist. TBrowserPane::PresentWord() wird aufgerufen, um die zwei API-Markierungen zusammen zu bringen, um die derzeitige Dateiposition wiederzugeben. Ein Aufruf wird dann gemacht an PauseFlashing(), um den Aufleuchtprozess zu unterbrechen und der derzeitige

Systemticker wird gespeichert in der Variablen pauseTick, die später benutzt wird, um die Anzahl an Takten zu berücksichtigen, die während dieser Pause verstrichen sind.

[0146] Falls im Falle keiner Pause die Benutzeranzeigeoption des oberen Ausschnitts aus ist (d. h. topOn ist FALSE) und es derzeit einen oberen Ausschnitt gibt (has TopPane ist TRUE), wird der obere Ausschnitt „entfernt" durch Aufruf von ReconfigureWindow(). TBrowserPane::BlockChanged() wird aufgerufen, um die derzeitige Textzeile wieder zu markieren, falls nötig. Um die Anzahl an Ticks von der WPM-Zählung auszuschließen, die während der Pause verstrichen sind, sollte der Ticker, um den WPM-Zähler zu starten (initialTick), vorgebracht werden. Dies wird erreicht, indem der derzeitige Tickzähler (calling GetTickCount()) genommen wird und der Tickzählerstand während der Pause (d. h. pauseTick) abgezogen wird. Die Variable pauseStarted wird dann zu TRUE gesetzt und FlashLoop() wird aufgerufen, um das Aufleuchten oder Aufblitzen erneut aufzunehmen.

46 CmResetWPM (Private Method)

```
void CmResetWPM();
```

[0147] Diese Funktion wird auf die Bitte hin aufgerufen, die Variablen, die eine Berechnung des WPM-Zählers enthalten, zurückzusetzen. Die Anzahl der so weit aufgeblitzten Worte wird zu Null gesetzt (d. h. wordCount) und der Ticker an den Startpunkt des ersten Flashs, da der Wortzähler Null auf den derzeitigen Takt mit GetTickCount() gesetzt wird.

47 CmHelpOnHelp (Private Method)

```
void CmHelpOnHelp();
```

[0148] Diese Funktion wird auf die Bitte hin aufgerufen, WINHELP.EXE aufzurufen mit der Hilfedatei WINHELP.HLP.

48 CmAbout (Private Method)

```
void CmAbout();
```

[0149] Diese Funktion wird aufgerufen auf die Bitte zum Aufruf eines „About"-Dialogs. Die Klasse TDelaysDialog 21c ist eine Dialogboxklasse und ist abgeleitet von TDialog. Dieses Objekt erhält die verschiedenen Verzögerungsraten vom Benutzer.

[0150] Die Elemente dieser Klasse sind wie folgt:

TDelaysDialog	***TDialog***
TDelaysDialog SetupWindow CanClose	 ~~SetupWindow~~ ~~CanClose~~

1 rates (Private Variable)

```
TCharonWin::TDelayRate& rates;
```

[0151] Dies ist ein Bezug zu der Strukturvariablen des Typs TDelaysDialog von seiner Mutter, die die verschiedenen Verzögerungsraten enthält (siehe ineinander gepackte Struktur TDelayRate innerhalb der TCharonWin-Klasse).

2 TDelaysDialog (Public Constructor)

```
TDelaysDialog(TWindow *parent, TCharonWin::TDelayRate& _rates)
```

[0152] Dieser Konstruktor erzeugt das Dialogobjekt. Der Parameter ist sein Vater-Zeiger und der Bezug zur Strukturvariablen rates in seiner Mutter.

3 SetupWindow (Private Function)

```
void SetupWindow();
```

[0153] Diese Funktion überschreibt die Funktion in der Stammklasse. Sie ruft SetDlgItemInt() auf, um die Variablen interval, blackout, punctuations, ends, cr in der ineinander gepackten Struktur TDelayRate zu setzen.

4 CanClose (Private Function)

```
BOOL CanClose();
```

[0154] Diese Funktion überschreibt die Funktion in der Stammklasse. Sie ruft GetDlgItemInt() auf, um den Inhalt ihrer Edit-Boxen in der variablen Struktur TDelayRate zu bekommen. Schließlich meldet sie den zurückgegebenen Wert von der Basis CanClose().
[0155] Die Klasse TFlashesDialog 21d ist eine Dialogboxklasse und ist abgeleitet von TDialog. Dieses Objekt erhält die Anzahl der Worte je Aufleuchten oder Blitz vom User.
[0156] Die Elemente dieser Klasse sind

TFlashesDialog	*TDialog*
TFlashesDialog SetupWindow CanClose	~~SetupWindow~~ ~~CanClose~~

1 numFlashes (Private Variable)

```
UINT& numFlashes;
```

[0157] Dies ist eine Referenz zu der Variablen in ihrer Mutter TCharonWin, die die Anzahl der Worte je Flash oder Aufleuchten enthält.

2 TFlashesDialog (Public Constructor)

```
TFlashesDialog(TWindow *parent, UINT& _numFlashes);
```

[0158] Dieser Konstruktor erzeugt das Dialogobjekt. Der Parameter ist sein Vater-Zeiger und die Referenz zur Variablen numFlashes in seiner Mutter.

3 SetupWindow (Private Function)

```
void SetupWindow();
```

[0159] Diese Funktion überschreibt die Funktion in der Stammklasse. Sie ruft SetDlgItemInt() auf, um die Variable numFlashes in der Editbox zu setzen, die im Resource Editor definiert ist.

4 CanClose (Private Function)

```
BOOL CanClose();
```

[0160] Diese Funktion überschreibt die Funktion in der Stammklasse. Falls die Basis CanClose() TRUE meldet, ruft sie GetDlgItemInt() auf, um den Inhalt ihrer Editbox in die Variable numFlashes zu bekommen. Die Funktion gibt TRUE zurück, d. h. ein Okay, um die Dialogbox zu schließen, wenn numFlashes größer als Null ist.
[0161] Die Klasse TBrowserPane 21e ist abgeleitet von TEditFile. Die Stammklasse enthält ein Editier-Fenster und kümmert sich um eine Blockauswahl deren Offset-Einstellungen durch Aufruf der API-Funktion GetSelection() erhalten werden kann. Diese werden hier als „API markers" bezeichnet.
[0162] TBrowserPane implementiert zwei zusätzliche Markierungen, die als „application markers" bekannt sind. Ihr Zweck besteht darin, die Offsets der Benutzerblockauswahl in Erinnerung zu behalten, die ursprünglich von den API-Markierungen notiert waren, so dass, obwohl die API-Markierungen zur Markierung einiger

anderer Blöcke verwendet werden, die ursprüngliche Auswahl noch festgehalten wird. Diese Klasse kümmert sich auch um das Zentrieren, wenn möglich, der Textzeile, die diese Untermarkierungen enthält.
[0163] Die Funktionen dieser Klasse sind wie folgt

TBrowserPane	*TEditFile*
TBrowserPane	
PresentLine	
GetSelectionStart	
BlockBegin	
BlockBegin	
BlockEnd	
BlockEnd	
IsSelectionChanged	
RememberBlock	
SelectBlock	
SelectAll	
SetupWindow	~~SetupWindow~~
EvSize	~~EvSize~~
CalcMiddleLine	

1 blkBegin (Private Variable)

```
UINT blkBegin
```

[0164] Dies ist die Anwendungsmarkierung am Anfang. Sie hält fest, wo der Benutzer gerne mit dem Lesen des Textes beginnen würde. Der Wert dieser Variablen wird durch das Mutter-Fenster TCharonWin mittels der Funktion RememberBlock() verändert, wenn sie erkennt, dass es eine explizite Änderung bezüglich der internen Markierungsabschnitte durch den Benutzer gibt entweder durch den Mousepointer oder den Tastaturcursor. Diese Detektion erfolgt nur, wenn die Anwendung versucht das Aufleuchten neu zu starten in TCharonWin::CmGo() und TCharonWin::Pause().

2 blkEnd (Private Variable)

```
UINT blkEnd;
```

[0165] Wie bei blkBegin ist dies eine Markierung der Endanwendung.

3 numberMidLines (Private Variable)

```
int numberMidLines;
```

[0166] Dies ist eine Hälfte der Anzahl an Textzeilen, die eine Instanz von TBrowserPane auf ihre derzeitigen Fenstergröße halten kann.

4 lastSelectionPos (Private Variable)

```
UINT lastSelectionPos;
```

[0167] Dies kennzeichnet den Anfangsoffset eines Aufleuchtens oder Blitzens von Text relativ zum Dateianfang. Sein Zweck besteht darin die Anfangs-API-Markierungen festzuhalten, während das Aufleuchten im Gange ist, so dass irgendeine geänderte Anordnung der Blockauswahl durch den Benutzer entweder durch den Mauszeiger oder den Tastaturcursor während des Aufleuchtvorgangs erkannt wird.

5 TBrowserPane (Public Constructor)

```
TBrowserPane (Twindow *parent);
```

[0168] Dies baut ein Browser-Fenster auf, das lastSelectionPos zu Null initialisiert. Sie berechnet auch die

Anzahl von Zeilen, die eine Hälfte des Fensters erlauben würde und speichert sie in der Variablen numberMidLines.

6 PresentLine (Public Function)

```
void PresentLine (UINT startPos, UINT endPos);
```

[0169] Diese Funktion hebt das Wort oder die Worte hervor, die in startPos und endPos enthalten sind und stellt den ausgewählten Textblock in die Mitte des Fensters wenn möglich.
[0170] Zuerst ruft die Funktion SetSelection() auf, um das Wort oder die Worte auszuwählen und damit hervorzuheben. Sie hält den Start der Auswahl in lastSelectionPos fest, so dass sie erkennen kann, dass der Benutzer ausdrücklich die Auswahl, falls eine vorhanden ist, neu angeordnet hat.
[0171] Um die Auswahl in die Mitte des Fensters zu stellen, fügt sie eine Hälfte der Anzahl von Zeilen im Fenster (d. h. numberMidLines) und die Zeilenzahl der ersten Zeile im Fenster (durch Aufruf GetFirstVisibleLine()) hinzu und erhält damit die absolute Zeilenzahl der mittleren Textzeile relativ zur gesamten Datei. Sie ruft GetLineFromPos() auf, um die Zeilenzahl zu erhalten, die ausgewählt wurde und zieht von dieser die absolute Zeilenzahl ab, um die Zahl von Zeilen zu erhalten, die gescrollt werden müssen, so dass die Auswahl in der Mitte des Fensters erscheint.

7 GetSelectionStart (Public Function)

```
UINT GetSelectionStart();
```

[0172] Diese Funktion gibt die Anfangs-API-Markierung zurück. Sie ruft die Basisfunktion GetSelection() auf, entfernt die Endmarkierung und meldet die Startmarkierung zurück.

8 BlockBegin (Public Function)

```
UINT BlockBegin();
```

[0173] Diese Überlast-Inlinefunktion meldet die Anwendungsmarkierung am Anfang blkBegin zurück.

9 BlockBegin (Public Function)

```
void BlockBegin (UINT val);
```

[0174] Diese Überlast-Inlinefunktion setzt die Anwendungsmarkierung am Anfang blkBegin zu dem Wert in val.

10 BlockEnd (Public Function)

```
UINT BlockEnd();
```

[0175] Diese Überlast-Inlinefunktion setzt die Anwendungsmarkierung am Ende blkEnd.

11 BlockEnd (Public Function)

```
void BlockEnd (UINT val)
```

[0176] Diese Überlast-Inlinefunktion setzt die Anwendungsmarkierung am Ende blkEnd zu dem Wert in val.

12 IsSelection Changed (Public Function)

```
BOOL IsSelectionChanged ();
```

[0177] Diese Inlinefunktion testet, ob der Benutzer den Cursor oder die Blockauswahl neu angeordnet hat, während das Aufleuchten, Blitzen oder Flashing im Gange ist. Dies erfolgt durch Testen, dass die Position, die derzeit durch die API-Markierung am Anfang während des Aufleuchtvorgangs ausgewählt wurde, dieselbe ist wie die, die zuvor in lastSelectionPos gespeichert wurde. Falls dieser Test TRUE ist, ist keine neue Positionierung erfolgt.

13 RememberBlock (Public Function)

```
void RememberBlock();
```

[0178] Diese Funktion erinnert sich an den derzeit ausgewählten Block, in dem die API-Blockmarkierungen in blkBegin und blkEnd gespeichert werden. Wenn beide API-Markierungen an derselben Position sind, dann ist kein Block im Fenster ausgewählt, nichts wird getan.

14 SelectBlock (Public Function)

```
void SelectBlock();
```

[0179] Diese Inlinefunktion ruft SetSelect() mit den Variablen blkBegin und blkEnd auf, um den gesamten Leseblock einzustellen. Damit werden den API-Markierungen dieselben Werte wie den Anwendungsmarkierungen gegeben.

15 SelectAll (Public Function)

```
void SelectAll();
```

[0180] Diese Funktion wählt den gesamten im Fenster enthaltenen Text durch Setzen von blkBegin auf das erste Zeichen (d. h. Null) und blkEnd auf das letztmögliche Zeichen (d. h. FFFF hex) aus. Sie ruft dann SetSelection(), um den Block zu setzen.

16 SetupWindow (Private Function)

```
void SetupWindow();
```

[0181] Diese Inlinefunktion überschreibt ihren Vorgänger. Ihr Zweck ist es ein Editieren innerhalb des Fensters unmöglich zu machen. Die Funktion ruft zuerst die Basis SetupWindow() und dann SetReadOnly() mit einem Parameter TRUE auf, um ein Schreiben unmöglich zu machen.

17 EvSize (Private Function)

```
void EvSize (UINT sizeType, TSize& size);
```

[0182] Diese Inlinefunktion überschreibt ihren Vorgänger. Ihr Zweck ist es, das Ereignis einer Fenstergrößenänderung aufzuspüren (d. h. WM SIZE) und CalcMiddleLine() aufzurufen. Sie erreicht dies, indem sie die Clientgröße des Fensters zwischen unten und oben nimmt und sie durch die Höhe des Fonts einschließlich der Lead-Zeilen teilt.

[0183] Die Klasse TFlashPane 21f wird abgeleitet von TWindow und verwaltet ergänzend die Anzeige des Wortes oder von Worten. Diese Textzeile und ihr Anzeigefont werden von ihrem Mutterfenster TCharonWin übernommen.

[0184] Die Elementfunktionen dieser Klasse sind wie folgt:

TFlashPane	***TWindow***
TFlashPane RepaintWindow Paint	~~Paint~~

1 parentWin (Private Variable)

```
TCharonWin *parentWin;
```

[0185] Dies ist der Zeiger zum Mutterfenster TCharonWin, von dem sie die Information über den anzuzeigenden Text und den Anzeigefont erhält.

2 TFlashPane (Public Constructor)

```
TFlashPane(TCharonWin *parent);
```

[0186] Dies baut ein Text-Flash-Fenster auf, in dem es parentWin mit der Adresse des Mutterfensters (d. h. TCharonWin) initialisiert.

3 RepaintWindow (Public Function)

```
void RepaintWindow();
```

[0187] Diese Inlinefunktion ruft Invalidate() und UpdateWindow() der Stammklasse auf, um das Clientfeld des Fensters sofort neu aufzuzeichnen.

4 Paint (Private Function)

```
void Paint(TDC& dc, BOOL, TRect&);
```

[0188] Diese Funktion überschreibt die Vorgängerfunktion, die immer aufgerufen wird, wenn das Fenster neu aufgebaut werden soll. Sie wählt den Mutteranzeigefont und gibt in strin von parentWin das Wort oder die Worte graphisch wieder.
[0189] Die Funktion bekommt zuerst ihre Fenster-Client-Dimensionen und bestimmt, dass der Font, der für dieses Fenster am besten geeignet wäre, halb so hoch ist wie die Höhe der Fensteranzeige (d. h. Fußzeile – Kopfzeile). Sie benutzt diesen Parameter, um den Anzeigefont von ihrer Hauptfunktion zu erhalten durch Aufruf von TCharonWin::SetDisplayFont(). Sie stellt die Texteinfügungsbasis auf das Zentrum des Texts, den sie anzeigt, durch Aufruf von SetTextAlign() mit TA CENTER), den Hintergrund dieses Font auf den Hintergrund des Fensters (durch Aufruf SetBKMode() mit TRANSPARENT) und die Farbe des Font auf jene, die im Mutterfont-Data.Color definiert ist. Sie fährt fort mit dem Schreiben des Text durch Aufruf Textout().
[0190] Schließlich wird die Klasse TString 21 g abgeleitet von der Stammklasse string, und ergänzend verwaltet sie das Lesen einer Anzahl von Worten (wie sie durch die Variable TCharonWin::numFlashes bestimmt wird) von einem Eingabestrom.
[0191] Die Elementfunktionen dieser Klasse sind wie folgt:

TString	***string***
GetAFlash GetAppendToken	

1 numFlashes (Public Variable)

```
UINT numFlashes;
```

[0192] Dies hält die Anzahl der Worte, wie vom Benutzer bestimmt, die vom Eingabedateistrom zu lesen sind. Die Anzahl der erfolgreich von dieser Klasse gelesenen Worte wird von der Klasse gehalten und letztendlich zur Anzeige im unteren Ausschnitt verwendet. Dies ist eine globale Variable und kann unmittelbar von Klasse TCharonWin verändert werden.

2 GetAFlash (Public Function)

```
int GetAFlash(istream _FAR &is, UINT &startPos, UINT &endPos);
```

[0193] Diese Funktion liest eine vom Benutzer spezifizierte Anzahl von Worten von einem Plattendateistrom und meldet die tatsächliche Anzahl von gelesenen Worten. Sie löscht zuerst ihren API-String-Speicher und ruft wiederholt GetAppendToken() auf, um ein Wort von einem Dateistrom istream(is) zu bekommen, bis die Anzahl (wie durch die Variable numFlashes bestimmt) erreicht ist. Die tatsächliche Anzahl an gelesenen Worten kann jeder Wert kleiner gleich dieser Variablen sein (siehe Elementfunktion GetAppendToken() für weitere Details). Die Startposition des Dateistroms, von dem das Lesen stattgefunden hat, wird in startPos zurückgegeben und die Endposition in endPos.

3 GetAppendToken (Private Function)

```
FLASH_STATUS GetAppendToken(istream _FAR &is);
```

[0194] Diese Funktion liest ein Wort und hängt es in seinen API-String-Speicher von einer Datei istream (is). Sie meldet einen Status UNSUCCESFUL bei einem erfolglosen Lesen, SUCCESFUL bei einem erfolgreichen Lesen und END OF_FLASH bei einem erfolgreichen Lesen, bei dem jedoch eine Punktuation oder eine Satzendmarkierung als letztes Zeichen des Wortes besteht. Der END OF_FLASH-Status informiert die Auffruftunktion (d. h. GetAFlash()), dass diese Funktion nicht länger aufgerufen werden soll ohne eine Bildschirmanzeige, da ihr interner Stringspeicher fertig ist.
[0195] Wenn ein Wort erfolgreich gelesen wurde, wird es ans Ende des internen String-Speichers angehängt. Falls der interne String-Speicher etwas enthält (d. h. ein früheres Wort), wird jedoch, bevor das Anhängen durchgeführt wird, ein Leerzeichen (d. h. ASCII 32) nach dem letzten Wort angehängt, bevor das jetzige Wort angefügt wird.
[0196] Ein Wort ist eine Zuordnungsmarkierung und ist definiert als Zeichen mit Leerzeichen vor und hinter dem Wort (siehe read token() der String-Klasse). Die Konstante PUNCTUATION_LIST in der globalen Funktion IsPunctuation() ist ein Satz von Punktuationen, die einen Satz nicht beenden und entweder den Anfang oder das Ende eines Wortes anzeigen. Die Konstante SENTENCE END LIST in der globalen Funktion IsSentenceEnd() ist ein Satz von Punktuationen, die das Satzende anzeigen.
[0197] Im Hinblick nun auf die Nicht-Klassen-Funktionen 23 sind diese wie folgt:

1 GetProfileInt

```
UINT GetProfileInt(char *keyName, UINT defVal);
```

[0198] Diese Funktion erhält einen Integerwert von der Datei CHARON.INI unter der Sektion [Charon Reader]. Der entryName ist der Name des Eintrags in CHARON.INI und defVal ist der Defaultwert oder Ausfallswert, falls ein derartiger Eintrag nicht besteht.
[0199] Sie ruft die API GetPrivateProfileInt().

2 WriteProfileInt

```
void WriteProfileInt(char *keyName, UINT val);
```

[0200] Diese Funktion schreibt einen Integerwert in die Datei CHARON.INI unter der Sektion [Charon Reader]. Der entryName ist der Name des Eintrags in CHARON.INI und val ist der Wert, den diese Funktion schreibt.
[0201] Ihr erster Aufruf wandelt den Wert in einen String um und ruft dann API WritePrivateProfileString().

3 Delay

```
void Delay(UINT factor)
```

[0202] Diese Funktion macht eine lokale Schleife von verschachtelten Statements solange eine Prozessorzeit nicht vorhanden ist. Der Parameter factor bestimmt die Anzahl der Schleifen, die diese Funktion machen muss. Diese Anzahl kann willkürlich sein, ihre Berechnung zu Beginn der Anwendung hilft jedoch die Anzahl zu bestimmen, die diese Funktion in einem Faktor von Millisekunden zum Laufen bringt (siehe TCharon::SetupWindow()).
[0203] Eine MS Windowsanwendung benutzt üblicherweise keine lokalen Verzögerungsschleifen (insbesondere ohne den Aufruf TApplication::PumpWaitingMessages()), da dieses Verfahren dazu führen würde, dass andere Windowsanwendungen und sie selbst einfriert. Diese Anwendung muss jedoch manchmal eine Takttickerauflösung feiner als 55 Millisekunden haben. Wenn eine solche Feinheit benötigt wird, wird diese Funktion verwendet (siehe TCharonWin::DelayUntilTime() für Details, wie Verzögerungen bedingt implementiert werden).

4 IsPunctuation

```
BOOL IsPunctuation(char mark);
```

[0204] Diese Funktion meldet TRUE, falls der Parameter mark ein Zeichen in der definierten Punktuationsliste

PUNCTUATION LIST ist.

5 IsSentenceEnd

```
BOOL IsSentenceEnd(char mark);
```

[0205] Diese Funktion meldet TRUE, wenn der Parameter mark ein Zeichen in der definierten Satzbeendigungsliste SENTENCE ENDLIST ist.
[0206] Aus der allgemeinen Beschreibung der Klassen und ihrer Elementfunktionen wird auf eine Anzahl von wichtigen Aspekten des Systems Bezug genommen, die besser aufgrund der Beschreibung des tatsächlichen Betriebs des Computerprogramms zu schätzen sind. Diese Beschreibung erfolgt nun unter Bezug auf das allgemeine Bildschirmlayout und aus der Nutzerperspektive des Betriebs des Computerprogramms. Demgemäß wird Bezug genommen auf das funktionale Flussdiagramm des Computerprogramms, das in **Fig.** 3 der Zeichnungen dargestellt ist, und die Bildschirmlayouts der Anzeige gemäß den **Fig.** 4 bis 18.
[0207] Zu Beginn wird das Computerprogramm durch Laufen lassen der ausführbaren Datei ‚.exe' ausgeführt, die durch das Programm und die dynamisch verbundenen Bibliotheken erzeugt wird, wie zuvor beschrieben. Dies ruft zu Beginn die TCharApp-Klasse von Funktionen 21a auf und erzeugt den Bildschirm, der in **Fig.** 4 der Zeichnungen gezeigt ist, durch Aufruf der Klasse TCharonWin an Funktionen 21b zusammen mit der Klasse TBrowserPane der Funktion 21e und der Klasse TFlashPane an Funktionen **21f**. Wie zu sehen ist, hat der Bildschirm das standardmäßige Windows (registrierte Marke) Layout mit einer Titelzeile **31**, einer Menüzeile 33, einer Werkzeugzeile 35, einem Browserausschnitt 37, einem Flash-Ausschnitt 39, einer Statuszeile **41**, einer Steuermenübox **43** und Tsatenfeldern zum Minimieren, Maximieren und Wiederherstellen, die allgemein bei **45** gezeigt sind.
[0208] Das Aufrufen dieser Initialisierungsfunktionen wird durch den Kreis 101 in **Fig.** 3 dargestellt.
[0209] Die Menüzeile umfasst die Menüüberschriften ‚File', ‚Speed', ‚Read', ‚Options' wie zuvor beschrieben. Die Werkzeugzeile umfasst ein Tastenfeld **47** neues Dokument, ein Startfeld **49**, ein Tastenfeld **41** zum Vergrößern der Scangeschwindigkeit, ein Tastenfeld **53** zum Verringern der Scangeschwindigkeit und ein Pausentastenfeld **55**.
[0210] Der Browserfensterausschnitt 37 umfasst einen vertikalen Scrollbalken **57** und einen horizontalen Scrollbalken **59**.
[0211] In der vorliegenden Ausführungsform ist der Browserausschnitt auf etwa 65% der Größe des Schirms eingestellt und der Flash-Ausschnitt auf etwa 35%.
[0212] Die Statuszeile **41** umfasst einen progressiven Zähler der Worte je Minute, die durch das Programm gescannt werden, und die Dauer des Zeitintervalls, über die ein diskreter Abschnitt des Texts angezeigt wird in Millisekunden.
[0213] Wie in **Fig.** 3 durch den Kreis 102 dargestellt, wird eine Datei durch Öffnen des ‚File'-Menüs ausgewählt, wie in **Fig.** 5 der Zeichnungen dargestellt, und Auswählen der ‚Open'-Option, was die ‚Open-Dialogbox' hervorbringt, wie in **Fig.** 6 der Zeichnungen dargestellt.
[0214] Es wäre anzumerken, dass die Mittel zum Verlassen des Programms auch unter dem ‚File'-Menü vorgesehen sind durch Auswählen der Exit-Option, was auch in **Fig.** 5 der Zeichnungen gezeigt ist.
[0215] In der ‚Open'-Dialogbox wird die Textdatei in bekannter Weise ausgewählt und im Fall des vorliegenden Ausführungsbeispiels wird die Datei ‚readtest.txt' ausgewählt.
[0216] Das Auswählen der gewünschten Datei ordnet zunächst den ersten größeren Textabschnitt davon in dem Browserausschnitt 37 an, wie in **Fig.** 7 der Zeichnungen dargestellt.
[0217] Vorbehaltlich der Einstellungen der Steuerparameter des Programms ist das Programm bereit, mit dem Lesen zu beginnen.
[0218] Das Einstellen der Steuerparameter wird durch den Kreis 103 in **Fig.** 3 dargestellt und es erfolgt mittels des Menüelements ‚Options'.
[0219] Wie in **Fig.** 8 der Figuren gezeigt, gibt es drei geeignete Optionen: nämlich ‚Delays', ‚Font' und ‚Number of Flashes', von denen jede eine ihr zugeordnete Dialogbox hat, und zwei Umschaltoptionen: nämlich ‚Auto size font' und ‚Browser window'.
[0220] Die erste Option ‚Delays' ruft die Klasse TDelaysDialog von Funktionen 21c auf und erzeugt eine Dialogbox, wie sie in **Fig.** 9 der Zeichnungen dargestellt ist, die die Steuerparameter listet: ‚Word Display Time', ‚Blank screen between words', ‚Punctuation', Sentence End' und ‚Paragraph End'.
[0221] Der Parameter ‚Word Display Time' bestimmt die Anzeigezeit oder -dauer, die ein diskreter Abschnitt eines angeordneten Textes an derselben Position im Flash-Ausschnitt 39 auf dem Bildschirm angezeigt wird, und dieser Wert ist derselbe Wert, der unter ‚Interval' in der Statuszeile **41** erscheint. Der Parameter ‚Blank screen between words' definiert das Ausblendezeitintervall, das zwischen der Anzeige von aufeinander folgenden Worten in dem Flash-Ausschnitt 39 erscheint. Der Parameter ‚Punctuation' definiert das Ausblendezeitintervall, das nach der Anzeige eines diskreten Zextabschnitts auf dem Flash-Ausschnitt bis zum nächsten dis-

kreten angezeigten Textabschnitt auftritt, wenn der erste Textabschnitt eine Punktuation wie ein Komma, Fragezeichen oder dergleichen umfasst. Der Parameter ‚Sentence End' definiert das Ausblendezeitintervall, das zwischen aufeinander folgenden, diskreten, anzuzeigenden Textabschnitten auftritt, wenn ein Satzende erreicht wird, das durch einen Punkt, ein Ausrufezeichen, ein Fragezeichen oder dergleichen gekennzeichnet ist. Schließlich definiert der Parameter ‚Paragraph End' das Ausblendezeitintervall, das nach der Anzeige eines diskreten Textabschnitts auftritt, der am Ende eines Absatzes erscheint, bis die Anzeige den nächsten diskreten Textabschnitt am Anfang des nächsten Absatzes anzeigt.

[0222] Die Option ‚Font' erzeugt die ‚Font'-Dialogbox, die in **Fig.** 10 der Zeichnungen gezeigt ist und die von einer der standardmäßigen dynamisch verlinkten Bibliotheken erhalten wird, wodurch der Font in bekannter Weise gewählt wird.

[0223] Die Option ‚Flashes' ruft, wenn sie ausgewählt wird, die Klasse TFlashesDialog an Funktionen 21d auf und bringt die in **Fig.** 11 der Zeichnungen gezeigte Dialogbox auf den Schirm. Diese ist vorgesehen zum Einstellen der Steuerparameter zum Bestimmen der Wortanzahl, die zu irgendeiner Zeit aufleuchten oder aufblitzen sollen, wobei es sich um die im diskreten, auf dem Flash-Ausschnitt 39 angeordneten Textabschnitt erscheinenden Worte handelt.

[0224] Hinsichtlich der Option ‚Auto size font' ruft deren Auswahl eine Funktion zur automatischen Größenauswahl des Fonts des angezeigten Textabschnitts im Flash-Ausschnitt 39 auf, wie es in **Fig.** 8 der Zeichnungen gezeigt ist. Im vorliegenden Ausführungsbeispiel springt der Font automatisch auf den Defaultwert der echten Type Times New Roman, die in einem regulären Font Stil einer Größe von 24 Pitch dargestellt wird. Um diese Einstellung zu ändern, ist es erforderlich, die Option ‚Auto size font' abzuwählen und neue Steuerparameter für den Font unter der Font-Option zu setzen, wie zuvor beschrieben.

[0225] Die Option ‚Browser window' ruft bei Auswahl eine Funktion auf, die dazu führt, dass der Browserausschnitt 37 als auch der Flash-Ausschnitt 39 gemäß **Fig.** 4 angezeigt wird in den zuvor beschriebenen Fensterverhältnissen. Wenn diese Funktion abgewählt wird, wird nur der Flash-Ausschnitt 39 dargestellt, der dann das vollständige Anzeigefenster einnimmt, wie in den **Fig.** 12 und 13 der Zeichnungen gezeigt. Diese Funktionen sind durch den Kreis 105 dargestellt.

[0226] Nach Einstellen der einstellbaren Steuerparameter kann das Leseprogramm dann aufgerufen werden, um mit dem Lesen des Textes zu beginnen. Dieser Programmaufruf kann durch das Menü ‚Read' ausgeführt werden, das Starttastenfeld **49** oder mittels eines Tastenbeschleunigers (Kurztaste). Im Fall des Menüs ‚Read' sieht das Pull-Down-Menü hierfür drei Optionen, wie in **Fig.** 14 der Zeichnungen dargestellt, vor, wobei die Erste ‚Go' ist, die Zweite ‚Restart' ist und die Dritte ‚Select all' ist. Auswählen der ‚Go'-Option oder Drücken der Kurztaste oder des Tastenbeschleunigers G oder einer entsprechenden Tastenkombination veranlasst die Funktion mit dem Lesen der Textdatei zu beginnen. Alternativ wird dieselbe Funktion durch Drücken der Enter-Taste oder durch Drücken des Starttastenfelds **49** aufgerufen.

[0227] Die Option ‚Restart' ist wirksam, nachdem das Programm mit dem Lesen begonnen hat, und führt nach Auswahl zu einer Funktion, um das Leseprogramm zu seinem Startstatus zurückzuführen und eine Eingabe zu erwarten. Diese Option kann auch durch Drücken des Tastenbeschleunigers ‚s' oder der Control- und Home-Tasten auf der Tastatur aufgerufen werden.

[0228] Die Option ‚Select all' ist der Default-Steuerparameter für die Blockbestimmung der zu lesenden Datei bei Initialisierung, wodurch der gesamte Text innerhalb der geöffneten Datei zum Lesen ausgewählt wird. Die Blockbestimmung wird kurz beschrieben. Der Zweck der Option ‚Select all' hat Wirkung, nachdem ein Textblock zum Lesen ausgewählt worden ist, und es erwünscht ist, zum Lesen der vollständigen Textdatei zurückzukehren. Diese Option kann auch durch Drücken des Tastenbeschleunigers oder der Kurztaste ‚S' aufgerufen werden.

[0229] Die Blockbestimmung wird unter der Klasse TBrowserPane an Funktionen 21e durchgeführt, wie sie durch den Kreis 104 gemäß **Fig.** 3 dargestellt ist. Die Blockbestimmung wie zuvor beschrieben hat eine Default-Einstellung, die die gesamte geöffnete Datei ist. Diese Funktion wird so unmittelbar nach Initialisierung aufgerufen. Die Blockbestimmung kann jedoch verändert werden, so dass nur ein bestimmter Absatz oder Satz der Datei gelesen wird, indem die Anfangs- und Endanwendungsmarkierungen für den angezeigten Text innerhalb des Browser-Ausschnittes gesetzt werden. Dies wird üblicherweise durch Verwendung der Maustaste durchgeführt, um den ‚I'-Strahlcursor an der gewünschten Startstelle des Textes anzuordnen, die linke Maustaste nach unten gedrückt gehalten wird und der ‚I'-Cursor bis zur Endposition des zu lesenden Textes gezogen wird. So wie der ‚I'-Strahl oder -Cursor gezogen wird, wird der ausgewählte Textblock fortschreitend hervorgehoben bis der gesamte Block definiert ist, wie in **Fig.** 15 der Zeichnungen gezeigt.

[0230] Die Anfangsanwendungsmarkierung zur Definition des Blocks kann neu positioniert werden, während die Testdatei gelesen wird, falls gewünscht, wie durch den Kreis 109 dargestellt. Ergänzend ruft das Programm, nachdem ein hervorgehobener Textblock vom Computerprogramm gelesen wurde, eine Funktion zum Hervorheben des Textblocks nach Erreichen der Endanwendungsmarkierung auf, wie durch den Kreis 116 repräsentiert.

[0231] In Ergänzung zu den einstellbaren Steuerparametern ist das Menü ‚Speed' zur Änderung von be-

stimmten Steuerparametern der Steuerparameter vorgesehen, nämlich zur Erhöhung und Verringerung der Lesegeschwindigkeit, wie durch den Kreis **111** repräsentiert, d. h. die Wortanzeigezeit und das Ausblendezeitintervall und das Zurücksetzen der Berechnung Wörter-je-Minute, wie durch den Kreis 112 repräsentiert, deren Ergebnis in der Statuszeile **41** des Bildschirms angezeigt wird, um die kommutative Worte-je-Minute-Leserate zurückzusetzen. Das dem ‚Speed'-Menüelement zugeordnete Pull-Down-Menü ist in **Fig.** 16 der Zeichnungen gezeigt.

[0232] Die ‚Increase'- und ‚Decrease'-Geschwindigkeitsoptionen sind jeweils mit den Tastenfeldern **51** bzw. **53** zum Erhöhen und Verringern der Geschwindigkeit verlinkt, um die relevanten in-line-Funktionen CmIncrease- bzw. CmDecrease-Rate aufzurufen, die die Funktion IncreaseAWord rufen, die allesamt zuvor im Hinblick auf die Klasse TCharonWin beschrieben wurden. Dem gemäß wird für geringe Anzeigezeiten durch Auswählen von ‚Increase' die ‚Word display time (Wortanzeigezeit)' um eine Millisekunde bei jeder Auswahl der ‚Increase'-Option oder bei Betätigen des Tastenfelds **51** zur Erhöhung der Geschwindigkeit verringert. Umgekehrt wird die ‚Word display time' um jeweils eine Millisekunde bei jeder Auswahl der Geschwindigkeitsoption ‚Decrease' oder bei Betätigung des Tastenfeldes **53** zur Verringerung der Geschwindigkeit erhöht. Alternativ wird bei hohen Anzeigezeiten die ‚Wortanzeigezeit' um eine Zeitperiode erhöht oder verringert, die einer 1 WPM-Änderung entspricht. Diese Funktionen werden durch den Kreis 113 repräsentiert. Im vorliegenden Ausführungsbeispiel ändern derartige Geschwindigkeitsänderungen die Steuerparameter für das Ausblendezeitintervall nicht, das mit dem gesetzten Wert konstant bleibt, wie er mittels der ‚Delays'-Dialogbox bestimmt wurde, wie in **Fig.** 9 der Zeichnungen gezeigt.

[0233] Die Option „update WPM" ruft die Funktion CmResetWPM auf, wie durch den Kreis 108 repräsentiert, um die Variablen zurückzustellen, die zur Worte je-Minute-Zählung gehören. Die Funktion ShowWordsPerMinute, die ihrerseits die Elementfunktion WordsPerMinute aufruft, von denen beide Elemente der TCharonWin-Klasse sind, werden im normalen Betriebsablauf des Computerprogramms aufgerufen und melden die derzeitige Zahl an Wörtern je Minute, die in der zuvor beschriebenen Weise berechnet ist, und zeigen dieselbe in der Statuszeile **41**. Da der Aufruf von CmResetWPM die Variablen zu Null zurückstellt, verändert sich die in der Statuszeile angezeigte WPM-Rate signifikant, so lange die Wortzählung nach dem Zurücksetzen niedrig ist.

[0234] Das Pull-Down-Menü für das Menü ‚Speed' umfasst auch die Option ‚Pause/Resume', die entsprechend mit dem Pausetastenfeld **55** verbunden ist. Auswählen der Option ‚Pause/Resume' oder der Pausetastenfeldumschalter zwischen dem Stoppen des Anzeigens von diskreten Textabschnitten zum Zeitpunkt des Anzeigens des Textes oder des Ausblendens zwischen Text und einem Starten der Anzeige oder des Aufleuchten bzw. Aufblitzens von diskreten Textabschnitten beabstandet durch das Ausblendezeitintervall erfolgt durch Aufruf der Funktion CmPause, die auch ein Element der Klasse TCharonWin ist, wie durch den Kreis **110** repräsentiert. Demgemäß kann das Lesen des Textes in dieser Weise zu jeder Zeit während des Lesevorgangs unterbrochen oder wiederaufgenommen werden.

[0235] Wie zuvor erwähnt, wird das Lesen eines definierten Textblockes durch Drücken des Starttastenfeldes **49**, Auswählen der ‚Go'-Option unter dem Menü ‚Read' oder durch Drücken der Enter-Taste auf der Tastatur begonnen. Jede dieser Aktionen ruft die Funktion CmGo auf, die auch ein Element der Klasse TCharonWin ist, die ihrerseits verursacht, dass andere Funktionen je nach Programmzustand aufgerufen werden. Damit ordnet die Funktion CmGo beim normalen Betrieb diskret und aufeinander folgende Textabschnitte an derselben Position auf dem Flash-Ausschnitt für eine vorgeschriebene Anzeigezeit an und von vorgeschriebenen Ausblendeezeitintervallen getrennt, welche Zeiten und Intervalle durch Einstellung der geeigneten, zuvor beschriebenen Steuertafeln bestimmt werden.

[0236] Für den Fall, dass der Browser-Ausschnitt 37 dahingehend ausgewählt wird, dass er gemeinsam mit dem Flash-Ausschnitt 39 angezeigt wird, wird der angezeigte diskrete Textabschnitt auf dem Flash-Ausschnitt gleichzeitig innerhalb des Browser-Ausschnitts hervorgehoben, wie in **Fig.** 17 der Zeichnungen mit dem Wort „maintaining" gezeigt. Die Funktion ruft auch Zentriermittel, dargestellt durch den Kreis 107, um jeden diskreten Textabschnitt lateral innerhalb des Flash-Ausschnitts so zu zentrieren, dass das Zentrum jedes diskreten Textabschnittes an derselben Stelle angezeigt wird. Ein Beispiel hiervon ist in den **Fig.** 12 und 13 gezeigt, wobei die Worte „the" und „developed" zentral in dem Flash-Ausschnitt angezeigt werden und auch in **Fig.** 17 mit dem Wort „maintaining" innerhalb des Flash-Ausschnitts.

[0237] Ferner ruft in dem Fall, wenn ausgewählt ist, dass der Browser-Ausschnitt angezeigt wird, die Funktion Zentrierscrollmittel auf, die durch den Kreis 106 repräsentiert sind, zum automatischen Scrollen der Präsentation des größeren Teils des im Browser-Ausschnitt angezeigten Texts und zum Zentrieren der Zeile des Textes, der dem angezeigten diskreten Textabschnitt im Flash-Ausschnitt entspricht, wie am besten in **Fig.** 18 gezeigt ist. Es ist jedoch anzumerken, dass ausreichend Textzeilen über der Zeile, die den entsprechenden diskreten Textabschnitt betont, wie in **Fig.** 18 gezeigt, innerhalb der Datei vorhanden sein müssen, damit die Zentrierungsscrollmittel mit einem automatischen Scrollen und Zentrieren der Zeile beginnen, die den Text präsentiert, der dem diskreten, im Browser-Ausschnitt angezeigten Textabschnitt entspricht. Wenn somit das Lesen von Text zu Beginn einer Datei beginnt, fängt der hervorgehobene Textabschnitt ausgehend von der obersten Zeile

an und bewegt sich fortschreitend im Browser-Ausschnitt nach unten, wie in **Fig.** 17 gezeigt, bis das Zentrum des Ausschnitts erreicht wird, bevor ein automatisches Scrollen und Zentrieren auftritt.

[0238] Es ist zu bemerken, dass beim Implementieren des Computerprogramms in der WINDOWS (3.1) -Umgebung (registrierte Marke) die Multitasking-Fähigkeit von WINDOWS berücksichtigt bzw. angepasst werden muss. In dieser Hinsicht müssen alle WINDOWS-Programme gemeinsam existieren und parallel mit anderen WINDOWS-Programmen arbeiten können, um das Auftreten eines Multitaskings zu ermöglichen. Konsequenterweise ist es nötig, dass Computerprogramm so aufzubauen, dass es innerhalb der Beschränkungen, die von WINDOWS auferlegt sind, bleibt, um WINDOWS die Durchführung eines Multitaskings von WINDOWS zu erlauben, indem es sein eigenes Zeitmanagement vornimmt.

[0239] Demgemäß umfasst das Computerprogramm seine eigenen Zeitmanagementmittel, die angerufen werden, wenn die Funktion DelayUntilTime während des Betriebs der Hauptfunktion FlashLoop des Programm aufgerufen wird, wobei beide Funktionen Teil der Klasse TCharonWin sind. Die Funktion erlaubt es dem Computerprogramm vielmehr, seinen eigenen Zeitmechanismuss zu verwenden, so dass die ‚Wortanzeigezeit' und das Ausblendezeitintervall auf einer hoch auflösenden Basis wie z. B. 1 Millisekunden-Inkremente oder-Dekremente angepasst werden kann, und es doch noch WINDOWS ermöglicht, sein eigenes Zeitmanagement mit anderen WINDOWS-Anwendungen durchzuführen, um ein Einfrieren oder Unmöglichmachen derselben zu vermeiden. In dieser Hinsicht überwachen die vorgenannten Funktionen die für das Durchführen von Aktionen innerhalb des Programms erforderliche Zeit zur Bestimmung, ob für derartige Aktionen eingeplant ist, dass sie mehr Zeit benötigen als das erlaubte erforderliche 55-Millisekunden-Intervall, bevor WINDOWS es erforderlich macht, einen Check für Aufrufe von anderen Anwendungen durchzuführen und auf deren Anforderungen antworten, wie dies durch den Kreis 115 repräsentiert ist. Wenn eine Aktion wie das Anzeigen eines Worts oder von Worten auf dem Bildschirm (d. h. die Wortanzeigezeit) oder das Leerlassen eines Blank-Screens zwischen Worten (d. h. das Ausblendezeitintervall) so eingeplant ist, dass sie größer als 55 Millisekunden ist, dann erlaubt das Programm WINDOWS, dass es eine Überprüfung auf Aufrufe durch andere Programme vornimmt und auf deren Anforderungen antwortet. Nachfolgend wird dann die Steuerung an das Computerprogramm nach 55 Millisekunden zurückgegeben und die verbleibende erforderliche Zeit zur Durchführung der bestimmten Aktion wird berechnet. Falls sie größer als 55 Millisekunden ist wiederholt sich der Zyklus.

[0240] Wenn die erforderliche Zeit dahingehend bestimmt wird, dass sie weniger als 55 Millisekunden ist, ruft die Funktion ein Unterprogramm auf, wodurch die Zeit bedingungslos auf Null verringert wird, wie durch den Kreis 114 repräsentiert. Dies ist möglich in Folge der vorherigen Berechnung, die bestimmt, wie viele Verringerungszyklen erforderlich waren, um eine Millisekunde an Zeit zu verspielen. Dieses Unterprogramm wird durch die Funktion Delay angerufen, die eine der zuvor beschriebenen Nicht-Klassen-Funktionen 23 ist. Nachdem die angemessene Zeit zu Null verringert worden ist, kehren die Funktionen zur Überwachung der Zeit zurück, die erforderlich ist, um die nächsten Aktionen innerhalb des Computerprogramms vorzunehmen.

[0241] Eine andere innerhalb des Computerprogramms vorgesehene Funktion ist das Ausblenden und Erneuern des Bildschirms zwischen der Anzeige von Worten.

[0242] Normalerweise hinterlässt die Darstellung eines Worts nach einem anderen ein Schattenbild, das Zeit benötigt, um angemessen zu verschwinden. Dies ist bei hohen Wort-je-Minute-Raten sehr auffällig. Dieses Bild wird entweder vom Auge verursacht und der Zeit, die erforderlich ist, ein Abbild des letzten Wortes zu löschen und das Abbild eines neuen Wortes anzunehmen oder möglicherweise als ein Resultat einer Bildschirmfunktion. Da sich ein Wortabbild auf dem Bildschirm „einbrennt", wird man es nicht insgesamt dadurch los, dass man nur damit aufhört es darzustellen, und das Abbild braucht Zeit, um zu verschwinden. Die Zeit zwischen den Worten muss ausreichend sein, um es dem Schattenabbild zu erlauben, angemessen zu verschwinden, oder sonst ist der allgemeine Eindruck verschwommen wie ein Schleier. Um daher das Problem zu verringern, wird eine Verzögerung mit einem leeren Bildschirm zwischen Wortdarstellungen eingeführt. Wenn das Ausblendezeitintervall anwächst, wird die Pause bemerkbar und das Gesamtbild ist mehr leer als notwendig. Das führt dazu, dass die Worte nicht so dunkel erscheinen wie sie für die beste Worterkennung sein sollten. Bei kleinen Ausblendezeitintervallen kann jedoch ein „happy medium" (guter Mittelweg) gefunden werden, der die Wortfolge ziemlich lesbar macht.

[0243] Konsequenterweise ruft das Computerprogramm die Funktion FlashLoop an, die einen Teil der Klasse TCharonWin bildet, um den gesamten Flash-Ausschnitt in derselben Farbe wie den Hintergrund wiederzugeben, unmittelbar nachdem die ‚Word display time' verstrichen ist, so dass der umgefärbte Ausschnitt während dem Ausblendezeitintervall angezeigt wird.

[0244] Es ist zu beachten, dass der Schutzbereich der vorliegenden Erfindung nicht auf die beschriebene besondere Ausführungsform beschränkt ist. Demgemäß sind Veränderungen und Verbesserungen der beschriebenen Ausführungsform der Erfindung ebenso vom Schutzbereich der Erfindung umfasst, der durch die anhängenden Ansprüche bestimmt wird. Derartige Variationen und Verbesserungen umfassen die folgenden Funktionen und/oder Merkmale:

i) Das Löschen eines bestimmten Textblocks aus dem Speicher auf andere Weise als durch Einsatz der ‚SelectAll'-Option;

ii) Den Browser-Ausschnitt zu einem skalierbaren Verhältnis des Schirms zu machen gegenüber dem Flash-Ausschnitt;
iii) Die Möglichkeit, eine Zielgeschwindigkeit einzugeben und das Computerprogramm zu veranlassen, auf die Bereitstellung dieser Rate hin zu arbeiten, d. h. Vorsehen einer Nennleserate;
iv) Auswählen verschiedener Verzögerungen oder Pausen am Ende eines Abschnitts, Kapitels oder dergleichen des Textes;
v) Die Möglichkeit mehrere Blöcke zur Auswahl und zum nachfolgenden Lesen zu bestimmen;
vi) Das Vorsehen, die verstrichene Zeit auf der Statuszeile anzuzeigen, seit der bestimmte Block oder die gesamte Lesesitzung begonnen wurde;
vii) Überwachung der exakten Stellung des diskreten Textabschnittes, der innerhalb eines Textblocks angezeigt wird, wie z. B. durch Zählen der Wortzahl, so dass eine genaue Worte je-Minute-Zahl unmittelbar nach Markierung neuer Blockdefinitionen und nach Lesebeginn desselben angezeigt werden kann, falls jenes Material schon gelesen worden ist;
viii) Berechnung und Anzeige der für das vollständige Lesen eines bestimmten Textblockes erforderlichen Zeit bei der derzeitigen Worte-je-Minute-Rate, die auf der Statuszeile angezeigt wird;
ix) Eine Möglichkeit für Bemerkungen, die entlang des Textes vorgesehen ist;
x) Funktionen wie z. B., Find' und ‚Go To'-Text in der Datei, wie vom Benutzer eingegeben;
xi) Lesen für eine vorgegebene Zeit, was die gleichzeitigen Funktionen eines Zählens der Anzahl der gelesenen Worte und der Worte-je-Minute-Rate für jene Zeitperiode durchführt;
xii) Die Fähigkeit die Taktrate während der Zeit zu unterbrechen, in der Titel gelesen werden oder auf Drop-Menüs zugegriffen wird;
xiii) Unterbrechen des Aufleuchtens von Text bei Symbolisierungen (iconisation);
xiv) Die Fähigkeit, das Lesen zu unterbrechen, wenn das Fenster für das Computerprogramm nicht das auf dem Bildschirm angezeigte Hauptfenster ist;
xv) Die Beschränkung bestimmter Eingabeparamter, um sicher zu stellen, dass die Werte innerhalb geeigneter Bereiche oder Zeichentypen fallen;
xvi) Die Fähigkeit unbegrenzte Dateitypen zu lesen;
xvii) Die Fähigkeit mehrere Dateiformate zu lesen, wie z. B. „WORDPERFECT" (Marke), „AMI-PRO" (Marke), „WORD" (Marke), etc;
xviii) Die Möglichkeit, das vollständige Computerprogramm als eine Funktion einzubinden , die innerhalb einer größeren Wortverarbeitungsanwendung angerufen wird;
xix) Die Fähigkeit, Text aufzugreifen und ihn von irgendwo zu lesen, wie z. B. vom WINDOWS (registrierte Marke) ‚Clipboard'-Programm;
xx) Ein Feature zum oberflächlichen Lesen;
xxi) Eine Standby-Möglichkeit (snooze facility);
xxii) Eine Möglichkeit für Stimmaktivierungen und -steuerungen; und
xxiii) Eine Möglichkeit, graphische Bilder zu behandeln mit Nutzeroptionen zur selbstständigen Darstellung oder zum selbstständigen Unterbrechen und zur Anfrage, ob angezeigt werden soll oder eine Anzeige ignoriert werden soll.

Patentansprüche

1. System (**11**) zum Darstellen von Text- und/oder graphischen Informationen zum Lesen, umfassend:
ein Anzeigemittel zum Anzeigen der Text- und/oder graphischen Informationen auf einem Anzeigemedium (**15**);
ein Verarbeitungsmittel (**13**) zum Empfangen der Text- und/oder graphischen Informationen in einer elektronisch codierten Form und zur Verarbeitung dieser zur Präsentation auf dem Anzeigemedium (**15**);
ein Steuermittel (**19**) zum Steuern der Präsentation der Text- und/oder graphischen Informationen auf dem Anzeigemedium (**15**); und
ein Eingabemittel (**17**) zum Eingeben von Steuersignalen zum Betrieb des Verarbeitungsmittels (**13**) und des Steuermittels (**19**) oder zum Verändern von Steuerparametern für das Verarbeitungsmittel (**13**) und das Steuermittel (**19**);
wobei das Steuermittel (**19**) eine Reihe von Steuerfunktionen (101 bis 116) umfaßt, die aufgerufen werden können, um sequentiell diskrete und sukzessive Teile der Text- und/oder graphischen Informationen mit einem vorbestimmten Ausblendzeitabstand an dieselbe Position auf dem Anzeigemedium (**15**) zu bringen, wobei jeder diskrete und sukzessive Teil eine vorbestimmte Zeit lang an dieser Position angezeigt wird; und
dadurch gekennzeichnet, daß eine der Steuerfunktionen (101 bis **116**) ein Zentrierungsmittel (107) zum lateralen Zentrieren jedes der Teile in der Position enthält.

2. System nach Anspruch 1, wobei das Steuermittel (**19**) ein Anwendungsmittel zum Definieren des allge-

meinen Layouts des Anzeigemediums (**15**) für die Präsentation der Text- und/oder graphischen Informationen und der visuellen Steuerattribute des Systems (**11**) durch die Steuerfunktionen (101 bis 116) enthält.

3. System nach Anspruch 2, wobei das Anwendungsmittel direkt auf die Steuerparameter für das Steuermittel (**19**) reagiert und die Steuerfunktionen zum Definieren und Spezifizieren bestimmter Eigenschaften der Präsentation der Text- und/oder graphischen Informationen in dem allgemeinen Layout gemäß der Einstellung eines bestimmten Steuerparameters dafür und der Eingabe der Steuersignale aus dem Eingabemittel (**17**) aufruft.

4. System nach Anspruch 2 oder 3, wobei das Steuermittel (**19**) eine Ausschnittmittelsteuerfunktion (105) zum Anordnen des allgemeinen Layouts zu einem oder mehreren Ausschnitten (37, 39), in denen die Text- und/oder graphischen Informationen präsentiert werden, enthält.

5. System nach Anspruch 4, wobei die Ausschnitte (37, 39) einen Flash-Ausschnitt (39) und einen Browser-Ausschnitt (37) enthalten, wobei der Flash-Ausschnitt (39) die Position und der Browser-Ausschnitt (37) einen Bereich, in den ein größerer Teil der Text- und/oder graphischen Informationen, aus dem die diskreten Teile abgeleitet werden, gebracht werden kann, enthält.

6. System nach Anspruch 5, wobei die Ausschnittmittelsteuerfunktion (105) dafür sorgt, daß der Flash-Ausschnitt (39) und der Browser-Ausschnitt (37) in Verbindung miteinander zum gleichzeitigen Betrieb (21b) in demselben Fenster angezeigt werden.

7. System nach Anspruch 5 oder 6, wobei das Steuermittel (**19**) eine Zentrierungsrollmittelsteuerfunktion (106) zum automatischen Scrollen der Präsentation des größeren Teils in dem Browser-Ausschnitt (37) und zum Zentrieren der Zeile des größeren Teils, die entsprechende Text- und/oder graphische Informationen für den diskreten Teil präsentiert, in dem Browser-Ausschnitt (37) umfasst.

8. System nach einem der Ansprüche 5 bis 7, wobei die Ausschnittmittelsteuerfunktion (105) ein Hervorhebungsmittel zum Hervorheben der entsprechenden Text- und/oder graphischen Informationen in dem Browser-Ausschnitt (37) enthält.

9. System nach einem der Ansprüche 5 bis 8, wobei das Steuermittel (**19**) eine Blockauswahlmittelsteuerfunktion (104) zum Auswählen eines Blocks der in dem Browser-Ausschnitt (37) angezeigten Text- und/oder graphischen Informationen zur nachfolgenden Präsentation der diskreten und sukzessiven Teile davon auf dem Flash-Ausschnitt (39) enthält.

10. System nach einem der vorhergehenden Ansprüche, wobei das Steuermittel (**19**) eine Verzögerungsmittelsteuerfunktion (112, 113) zum Einstellen der Steuerparameter für die vorbestimmte Anzeigezeit enthält.

11. System nach einem der vorhergehenden Ansprüche, wobei das Steuermittel (**19**) eine Teilauswahlmittelsteuertunktion (103) zum Einstellen der Steuerparameter für die diskreten und sukzessiven Teile enthält.

12. System nach einem der vorhergehenden Ansprüche, soweit sie von von Anspruch 6 abhängig sind, wobei das Steuermittel (**19**) eine Ausschnittskalierungsmittelsteuertunktion enthält, durch die der Browser-Ausschnitt (37) in einen skalierbaren Anteil eines sowohl den Browser-Ausschnitt (37) als auch den Flash-Ausschnitt (39) enthaltenden Fensters, relativ zu dem Flash-Ausschnitt (39), verwandelt werden kann.

13. System nach Anspruch 2 oder einem der Ansprüche 3 bis 12 , soweit sie von von Anspruch 2 abhängig sind, wobei das Steuermitel (**19**) eine Scanratenmittelsteuerfunktion (112) zum Berechnen einer progressiven Zählung der pro Zeitintervall angezeigten Text- und/oder graphischen Informationen und zum Anzeigen der progressiven Zählung auf dem allgemeinen Layout enthält.

14. System nach Anspruch 13, wobei die Scanratenmittelsteuerfunktion (112) die Wort-pro-Minute-Rate von an die Position gebrachten und dort angezeigten Textinformationen berechnet und anzeigt.

15. System nach einem der vorhergehenden Ansprüche, wobei das Steuermittel (**19**) eine Steuerparameterveränderungsmittel-steuerfunktion (111) zum Verändern bestimmter der einstellbaren Steuerparameter (112, 113) auf gleichzeitige und von dem Betrieb der anderen Steuerparameter und Funktionen des Steuermittels (**19**) unabhängige Weise enthält.

16. System nach Anspruch 15, wobei die Steuerparameterveränderungsmittelsteuerfunktion (111) mit der Verzögerungsmittelsteuerfunktion (112, 113) in Wechselwirkung tritt, um die Steuerparameter für die vorbestimmte Zeitverzögerung, nachdem sie eingestellt wurden, auf gesteuerte und kontinuierliche Weise zu verändern.

17. System nach einem der vorhergehenden Ansprüche, wobei das Steuermittel (**19**) eine weitere Verzögerungsmittelsteuerfunktion (112, 113) zum Einstellen der Steuerparameter für die vorbestimmte Ausblendzeit enthält.

18. System nach Anspruch 17, soweit er von von Anspruch 15 oder 16 abhängig ist, wobei die Steuerparameterveränderungsmittelsteuerfunktion (111) mit der weiteren Verzögerungsmittelsteuerfunktion (112, 113) in Wechselwirkung tritt, um die Steuerparameter für die vorbestimmte Ausblendzeit, nachdem sie eingestellt wurden, auf gesteuerte und kontinuierliche Weise zu verändern.

19. System nach Anspruch 4 oder einem der Ansprüche 5 bis 18, soweit sie von von Anspruch 4 abhängig sind, wobei das Steuermittel (**19**) eine Umfärbemittelsteuerfunktion zum Umfärben des Ausschnitts (39), auf den die Textund/oder graphischen Informationen sequentiell gebracht werden, mit derselben Farbe wie die Hintergrundfarbe des Ausschnitts (39) für die gesamte Dauer der vorbestimmten Ausblendzeit unmittelbar nach dem Ablaufen der vorbestimmten Anzeigezeit (21b) enthält.

20. System nach einem der vorhergehenden Ansprüche, wobei das Verarbeitungsmittel (**13**) ein Mikrocomputer mit einer graphischen Benutzeroberfläche mit einer Multitasking-Fensterbetriebsfunktion (115) ist.

21. System nach Anspruch 20, wobei das Steuermittel (**19**) ein Zeitmanagementmittel (114) zum Überwachen der voraussichtlichen und vergangenen Zeit von dadurch ausgeführten Funktionen und zum Koordinieren der Abgabe und Rückgabe der Steuerung zu und von der Multitasking-Fensterbetriebsfunktion (115) enthält.

22. System nach Anspruch 21, wobei das Zeitmanagementmittel (114) die Zeit überwacht, die erforderlich ist, um Funktionen des Systems auszuführen, und bestimmt, ob für eine aktuelle Funktion eingeplant ist, daß sie mehr Zeit als das von der Multitasking-Fensterbetriebsfunktion (115) für solche Funktionen erlaubte erforderliche Zeitintervall in Anspruch nehmen kann; wobei das Zeitmanagementmittel (114) als Reaktion auf eine Bestimmung von dafür eingeplanten Funktionen, mehr Zeit als das erforderliche Zeitintervall in Anspruch zu nehmen, die Steuerung des Systems an die Multitasking-Fensterbetriebsfunktion (115) abgibt und die Steuerung wiedergewinnt, nachdem sie von der Multitasking-Fensterbetriebsfunktion (115) zurückgegeben wird, und das Zeitmanagementmittel (114) nach der Rückgabe die verbleibende Zeit zur Ausführung der aktuellen Funktion berechnet und, wenn sie größer als die erforderliche Zeitfunktion ist, das Abgeben der Steuerung an die Multitasking-Fensterbetriebsfunktion (115) wiederholt; und das Zeitmanagementmittel (114) als Reaktion auf eine Bestimmung von dafür eingeplanten Funktionen, weniger Zeit als das erforderliche Zeitintervall in Anspruch zu nehmen, bedingungslos die Zeit verringert, bis die Funktion abgeschlossen ist, und die nächste Funktion überwacht.

23. System nach einem der Ansprüche 20 bis 22, wobei das Steuermittel (**19**) ein Computerprogramm ist, das eine Kompilierung mehrerer Quelldateien umfaßt, darunter einer Hauptquellcodedatei, einer Kopfdatei, einer Betriebsmitteldatei und einer Betriebsmittelkopfdatei, wobei die Quelldateien eine ausführbare Datei bilden, die zusammen mit einer oder mehreren vorbestimmten, dynamisch angeknüpften Bibliotheken auf dem Computer ausgeführt werden kann.

24. System nach einem der vorhergehenden Ansprüche, wobei das Steuermittel (**19**) ein Computerprogramm ist, das über Menüelemente betrieben wird, die jeweils einen entsprechenden Beschleuniger zum Heraussenden einer Nachricht, die eine Funktion in einer vorbestimmten Klasse aufruft, aufweisen, wobei die vorbestimmten Klassen eine oder mehrere der folgenden Funktionen enthalten:
i) Erzeugung (21a) einer Anwendung für das Computerprogramm und einer Instanz für die Funktion, die den Browser-Ausschnitt (37) und den Flash-Ausschnitt (39) hält;
ii) Halten (21b) des Browser-Ausschnitts (37) und des Flash-Ausschnitts (39) zur Anzeige auf dem Anzeigemedium (**15**);
iii) Empfangen (21c) verschiedener Verzögerungsraten für den Betrieb des Flash-Ausschnitts (39) von dem Eingabemittel (**17**);
iv) Empfangen (21d) der Anzahl von diskreten Elementen der Text- und/oder graphischen Informationen pro Flash des Flash-Ausschnitts (39) von dem Eingabemittel (**17**);
v) Speichern (21e) der Offsets, die die Blockauswahl der Text- und/oder graphischen Informationen identifizie-

ren, und Zentrieren dieser in dem Browser-Ausschnitt (37);
vi) Verwalten (21f) der Anzeige der Text- und/oder graphischen Informationen in dem Flash-Ausschnitt (39);
vii) Anzeigen (21g) einer vorbestimmten Anzahl von diskreten Elementen der Text- und/oder graphischen Informationen, um den Teil zu bilden.

25. System nach Anspruch 24, wobei die Funktion weitere Funktionen aufrufen kann, die den von dem Steuermittel (**19**) nach einem der vorhergehenden Ansprüche bereitgestellten Funktionen entsprechen, wobei die Funktionen das Zentrierungsmittel (107), das Anwendungsmittel, das Ausschnittmittel (105), das Zentrierungsrollmittel (106), das Hervorhebemittel, das Verzögerungsmit tel und das weitere Verzögerungsmittel (112, 113), das Teilauswahlmittel (**103**), das Ausschnittskalierungsmittel, das Scanratenmittel (112), das Steuerparameterveränderungsmittel (111), das Flash-Ausschnittumfärbemittel und das Zeitmanagementmittel (114) umfassen.

26. Verfahren zum Darstellen von Text- und/oder graphischen Informationen zum Lesen, mit den folgenden Schritten:
Empfangen der Text- und/oder graphischen Informationen in einer elektronisch codierten Form;
Verarbeiten dieser zur Präsentation auf einem Anzeigemedium;
Auswählen diskreter und sukzessiver Teile der Text- und/oder graphischen Informationen;
sequentielles Anzeigen der Teile an derselben Position des Anzeigemediums (**15**) mit einem vorbestimmten Ausblendzeitabstand;
Anzeigen jedes diskreten und sukzessiven Teils an der Position für eine vorbestimmte Anzeigezeit; und
gekennzeichnet durch
laterales Zentrieren jedes der Teile in der Position.

27. Verfahren nach Anspruch 26, mit dem Schritt des Definierens und Spezifizierens bestimmter Eigenschaften der Präsentation der Text- und/oder graphischen Informationen in einem allgemeinen Layout zur Anzeige auf dem Anzeigemedium (**15**) gemäß der Einstellung bestimmter Steuerparameter dafür.

28. Verfahren nach Anspruch 27, mit dem Schritt des Anordnens des allgemeinen Layouts zu einem oder mehreren Ausschnitten (37, 39), in denen die Textund/oder graphischen Informationen vorgeschrieben werden.

29. Verfahren nach Anspruch 28, wobei die Ausschnitte (37, 39) einen Flash-Ausschnitt (39) und einen Browser-Ausschnitt (37) enthalten, wobei der Flash-Ausschnitt (39) die Position und der Browser-Ausschnitt (37) einen Bereich, in den ein größerer Teil der Text- und/oder graphischen Informationen, aus dem die diskreten Teile abgeleitet werden, gebracht werden kann, enthält.

30. Verfahren nach Anspruch 29, mit dem Schritt des Anzeigens des Flash-Ausschnitts (39) und des Browser-Ausschnitts (37) in demselben Fenster in Verbindung miteinander und unter gleichzeitigem Betrieb derselben (21 b).

31. Verfahren nach Anspruch 29 oder 30, mit dem Schritt des automatischen Scrollens der Präsentation des größeren Teils in dem Browser-Ausschnitt (37) und des Zentrierens der Zeile des größeren Teils, die entsprechende Textund/oder graphische Informationen für den diskreten Teil präsentiert, in dem Browser-Ausschnitt (37).

32. Verfahren nach einem der Ansprüche 29 bis 31, mit einem Hervorhebungsmittel zum Hervorheben der entsprechenden Text- und/oder graphischen Informationen in dem Browser-Ausschnitt (37).

33. Verfahren nach einem der Ansprüche 29 bis 32, mit dem Schritt des Auswählens eines Blocks der in dem Browser-Ausschnitt (37) angezeigten Textund/oder graphischen Informationen (21e) und des nachfolgenden Präsentierens der diskreten und sukzessiven Teile davon (21b) auf dem Flash-Ausschnitt (39).

34. Verfahren nach einem der Ansprüche 26 bis 33, mit dem Schritt des Einstellens der Steuerparameter für die vorbestimmte Anzeigezeit (21c).

35. Verfahren nach einem der Ansprüche 26 bis 34, mit dem Schritt des Einstellens der Steuerparameter für den diskreten und sukzessiven Teil (103).

36. Verfahren nach einem der Ansprüche 26 bis 35 , soweit sie von von Anspruch 30 abhängig sind, mit

dem Schritt des Verwandelns des Browser-Ausschnitts (**37**) in einen skalierbaren Anteil des allgemeinen Layouts relativ zu dem Flash-Ausschnitt (39).

37. Verfahren nach Anspruch 27 oder einem der Ansprüche 28 bis 36, soweit sie von von Anspruch 27 abhängig sind, mit dem Schritt des Berechnens einer progressiven Zählung der pro Zeitintervall angezeigten Text- und/oder graphischen Informationen und des Anzeigens der progressiven Zählung auf dem allgemeinen Layout. (**Fig.** 4).

38. Verfahren nach einem der Ansprüche 26 bis 37, mit dem Schritt des Veränderns bestimmter der einstellbaren Steuerparameter auf gleichzeitige und von dem Betrieb anderer Steuerparameter und Funktionen unabhängige Weise.

39. Verfahren nach Anspruch 38, mit dem Schritt des Veränderns der Steuerparameter für die vorbestimmte Verzögerungszeit, nachdem sie eingestellt wurden, auf gesteuerte und kontinuierliche Weise.

40. Verfahren nach einem der Ansprüche 26 bis 39, mit dem Schritt des Einstellens der Steuerparameter für die vorbestimmte Ausblendzeit.

41. Verfahren nach Anspruch 40, soweit er von von Anspruch 38 oder 39 abhängig ist, mit dem Schritt des Veränderns der Steuerparameter für die vorbestimmte Ausblendzeit, nachdem sie eingestellt wurden, auf gesteuerte und kontinuierliche Weise (21b).

42. Verfahren nach Anspruch 28 oder einem der Ansprüche 29 bis 41, soweit sie von von Anspruch 28 abhängig sind, mit dem Schritt des Umfärbens des Ausschnitts (39) mit derselben Farbe wie die Hintergrundfarbe des Flash-Ausschnitts (39) für die gesamte Dauer der vorbestimmten Ausblendzeit unmittelbar nach dem Ablaufen der vorbestimmten Anzeigezeit.

43. Verfahren nach einem der Ansprüche 26 bis 42, mit dem Schritt des Präsentierens der Text- und/oder graphischen Informationen zum Lesen in Verbindung mit einer Multitasking-Fensterbetriebsfunktion (115).

44. Verfahren nach Anspruch 43, mit dem Schritt des Überwachens der voraussichtlichen und vergangenen Zeit von durch das Präsentieren der Textund/oder graphischen Informationen ausgeführten Funktionen und des Koordinierens der Abgabe und Rückgabe der Steuerung zu und von der Multitasking-Fensterbetriebsfunktion (115).

45. Verfahren nach Anspnruch 44, mit dem Schritt des Überwachens der Zeit, die erforderlich ist, um Funktionen auszuführen, und des Bestimmens, ob für eine aktuelle Funktion eingeplant ist, daß sie mehr Zeit als das von der Multitasking-Fensterbetriebsfunktion (115) für solche Funktionen erlaubte erforderliche Zeitintervall in Anspruch nehmen kann, des Abgebens der Steuerung an die Multitasking-Fensterbetriebsfunktion (115) und des Wiedergewinnens der Steuerung von dieser als Reaktion auf eine Bestimmung einer dafür eingeplanten aktuellen Funktion, mehr Zeit als das erforderliche Zeitintervall in Anspruch zu nehmen, des Berechnens der verbleibenden Zeit zur Ausführung der aktuellen Funktion nach der Rückgabe der Steuerung von der Multitasking-Fensterbetriebsfunktion (115) und, wenn sie größer als die erforderliche Zeitfunktion ist, des Wiederholens des Abgebens der Steuerung an die Multitasking-Fensterbetriebsfunktion (115) und des bedingungslosen Verringerns der Zeit, bis die Funktion abgeschlossen ist, als Reaktion auf die Bestimmung einer dafür eingeplanten aktuellen Funktion, weniger Zeit als das erforderliche Zeitintervall in Anspruch zu nehmen, und des Überwachens der nächsten Funktion nach dem Verringern.

46. Verfahren nach einem der Ansprüche 26 bis 45, mit dem Schritt des Betreibens von Menüelementen, die jeweils einen entsprechenden Beschleuniger zum Aussenden einer Nachricht zum Aufruf einer Funktion in einer vorbestimmten Klasse aufweisen, wobei die vorbestimmten Klassen eine oder mehrere der folgenden Funktionen enthalten:
i) Erzeugung (21a) einer Anwendung für das Computerprogramm und einer Instanz für die Funktion, die den Browser-Ausschnitt (37) und den Flash-Ausschnitt (39) hält;
ii) Halten (21b) des Browser-Ausschnitts (37) und des Flash-Ausschnitts (39) zur Anzeige auf dem Anzeigemedium (**15**);
iii) Empfangen (21c) verschiedener Verzögerungsraten für den Betrieb des Flash-Ausschnitts (39) von dem Eingabemittel (**17**);
iv) Empfangen (21d) der Anzahl von diskreten Elementen der Text- und/oder graphischen Informationen pro Flash des Flash-Ausschnitts (39) von dem Eingabemittel (**17**);

v) Speichern (21e) der Offsets, die die Blockauswahl der Text- und/oder graphischen Informationen identifizieren, und Zentrieren dieser in dem Browser-Ausschnitt (37);
vi) Verwalten (21f) der Anzeige der Text- und/oder graphischen Informationen in dem Flash-Ausschnitt (39);
vii) Anzeigen (21g) einer vorbestimmten Anzahl von diskreten Elementen der Text- und/oder graphischen Informationen, um den Teil zu bilden.

Es folgen 18 Blatt Zeichnungen

Anhängende Zeichnungen

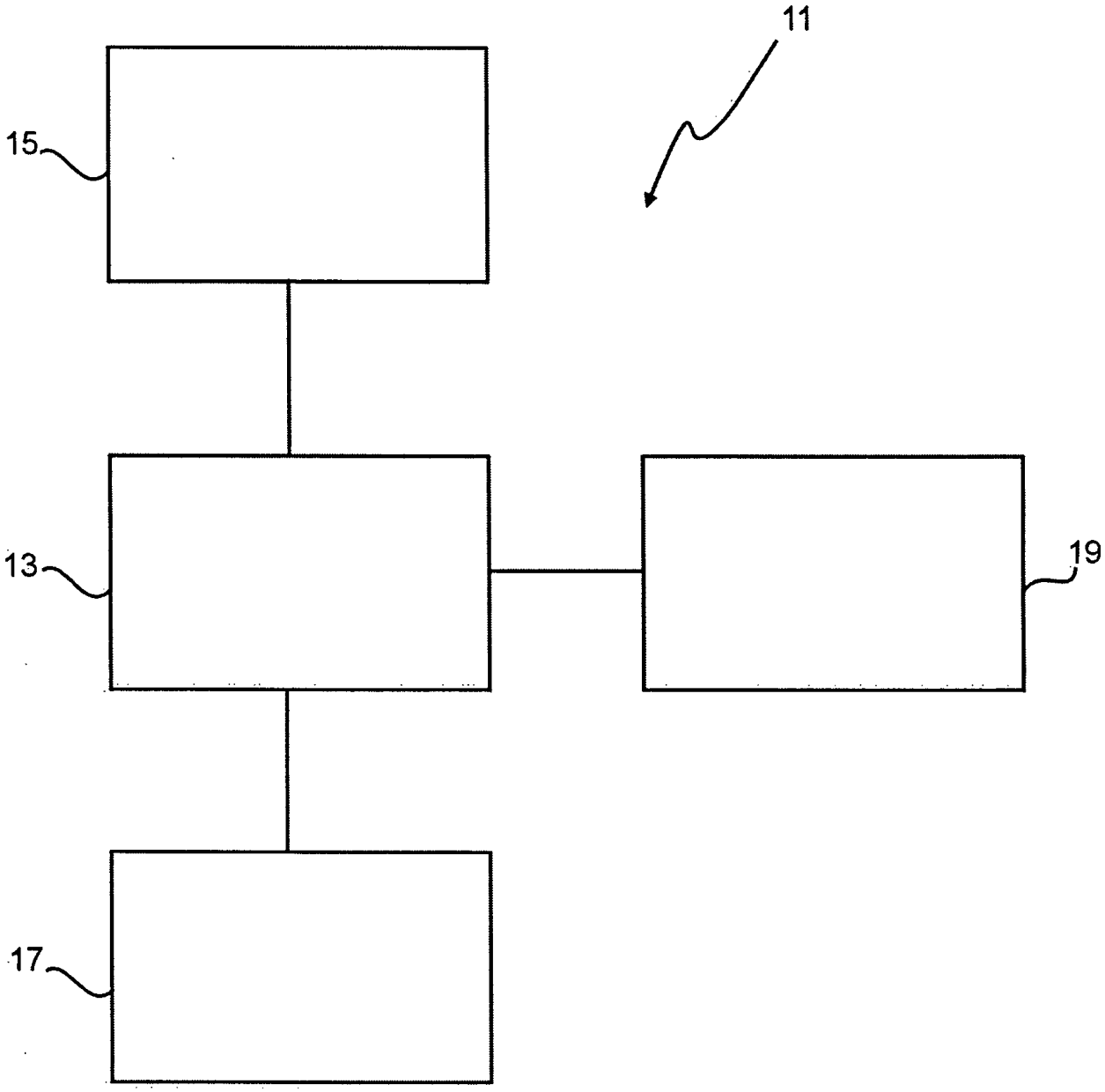


Fig. 1

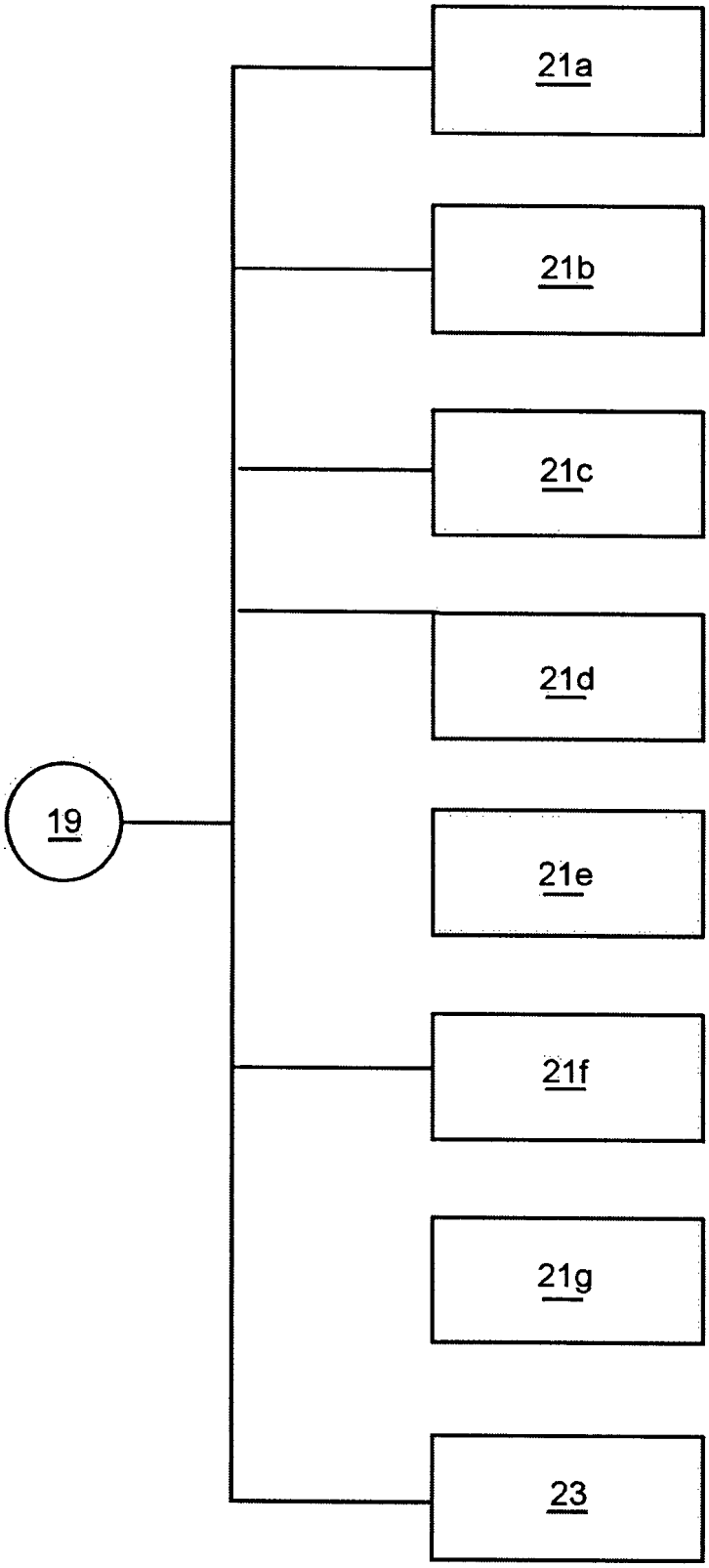


Fig. 2

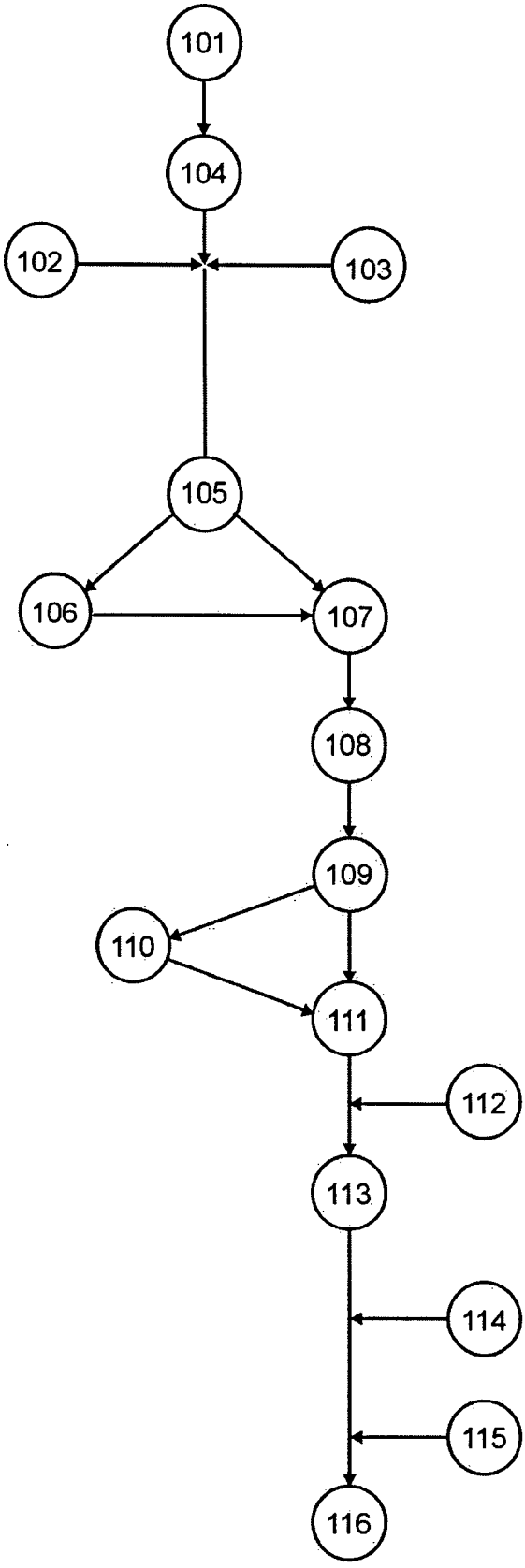


Fig.3

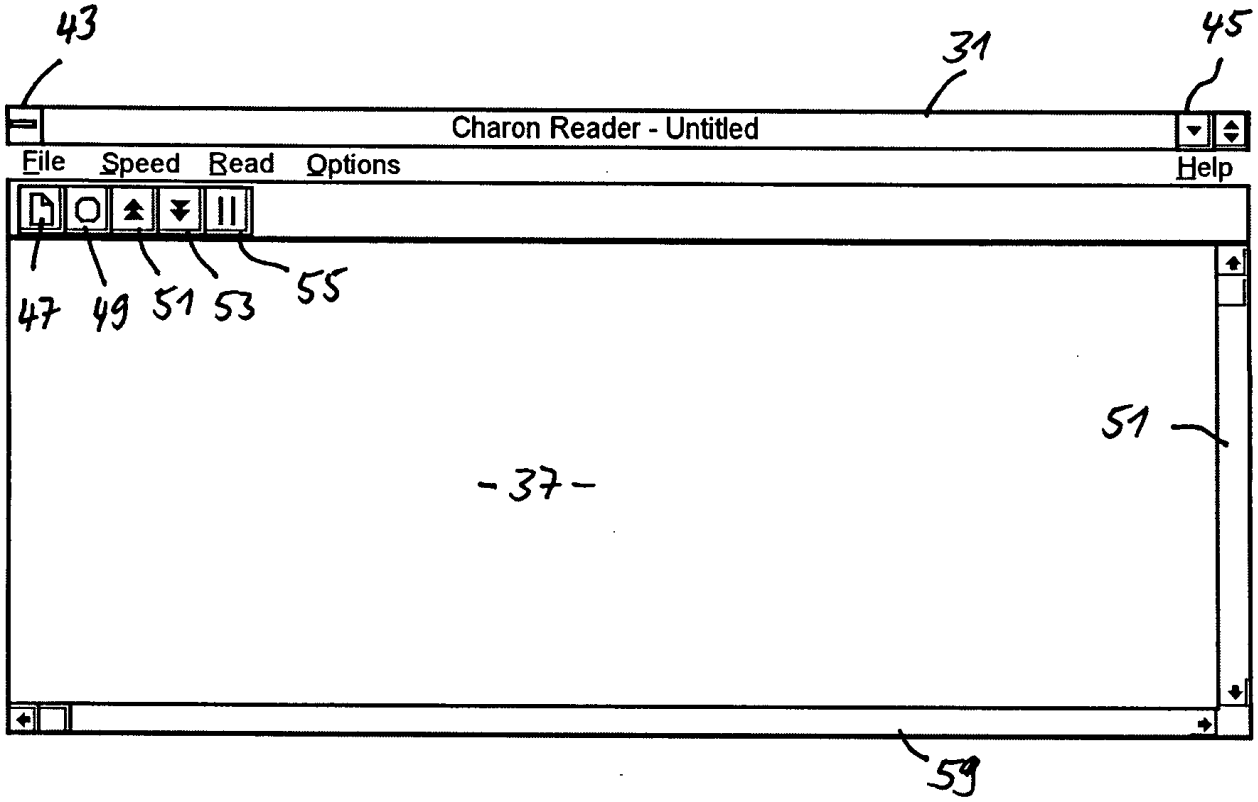

43
31
45
Charon Reader - Untitled
File
Speed
Read
Options
Help
47
49
51
53
55
51
- 37 -
59
- 39 -
Words Per Minute : Interval : 400
41

Fig. 4

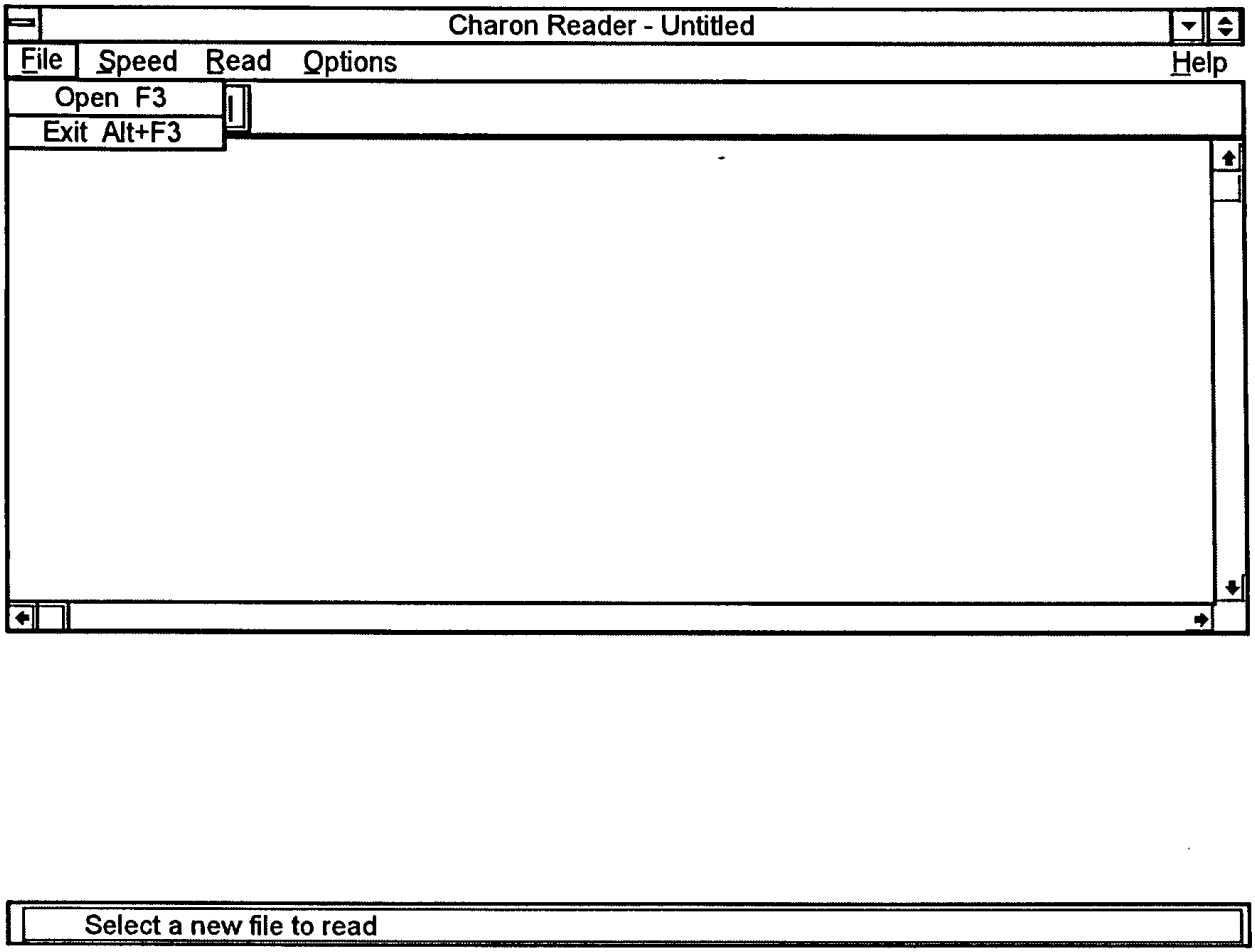

Charon Reader - Untitled
File
Speed
Read
Options
Help
Open F3
Exit Alt+F3
Select a new file to read

Fig. 5

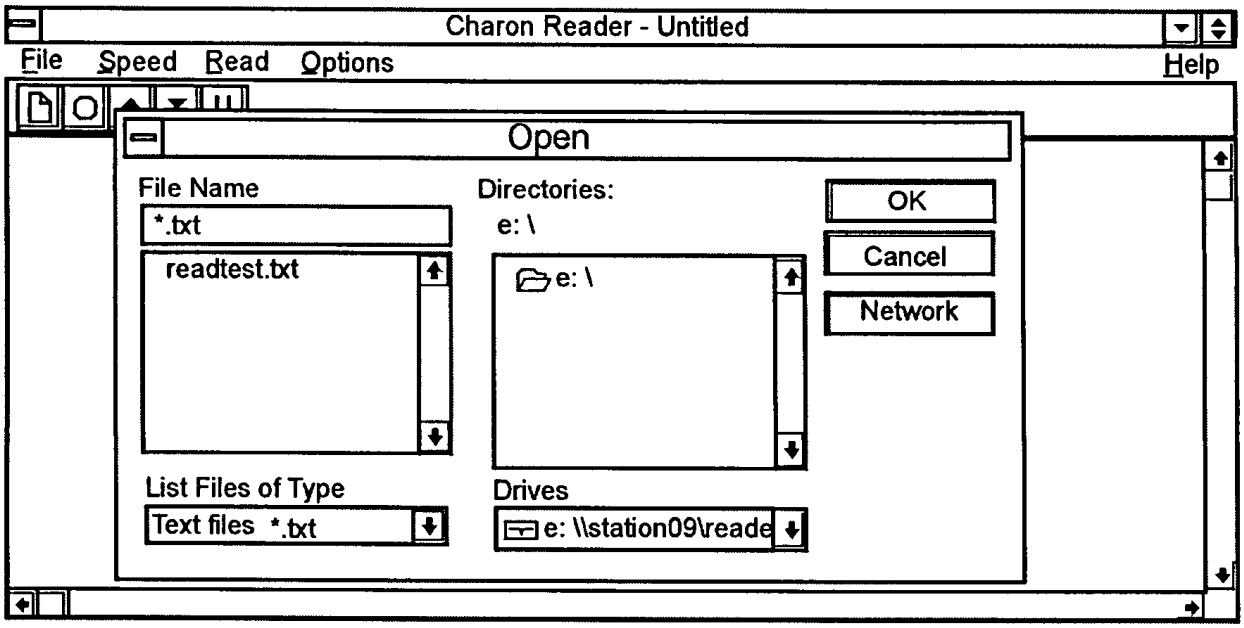

Charon Reader - Untitled
File
Speed
Read
Options
Help
Open
File Name
*.txt
readtest.txt
Directories:
e: \
e: \
OK
Cancel
Network
List Files of Type
Text files *.txt
Drives
e: \\station09\reade
Words Per Minute : 0 Interval : 400

Fig.6

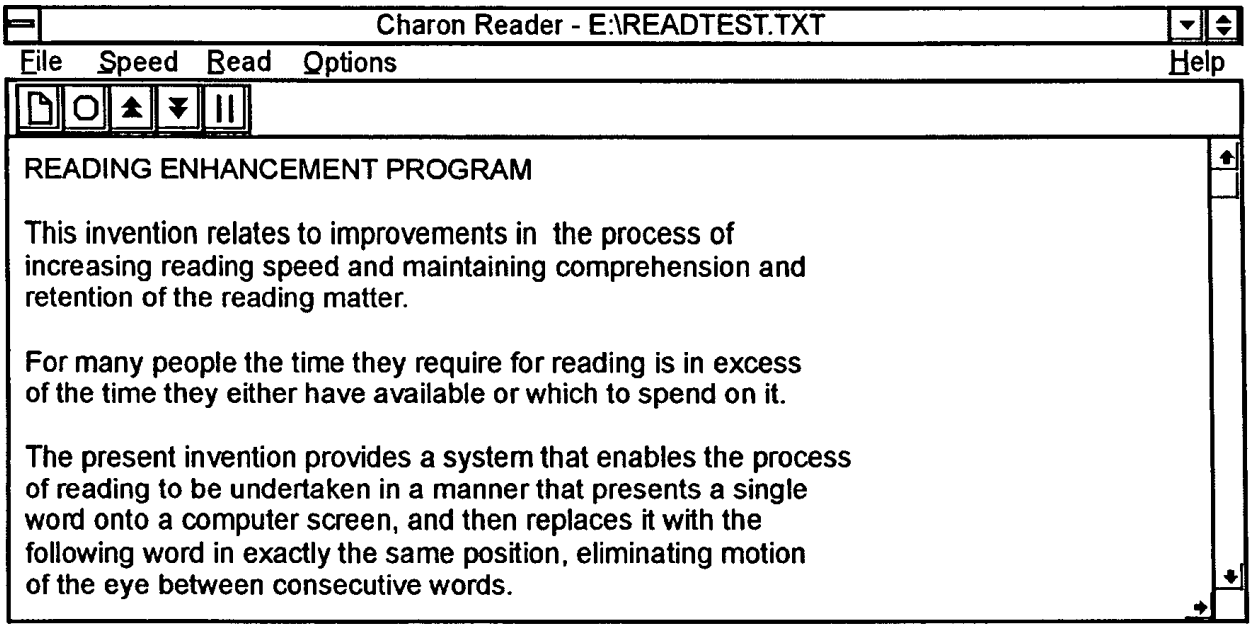
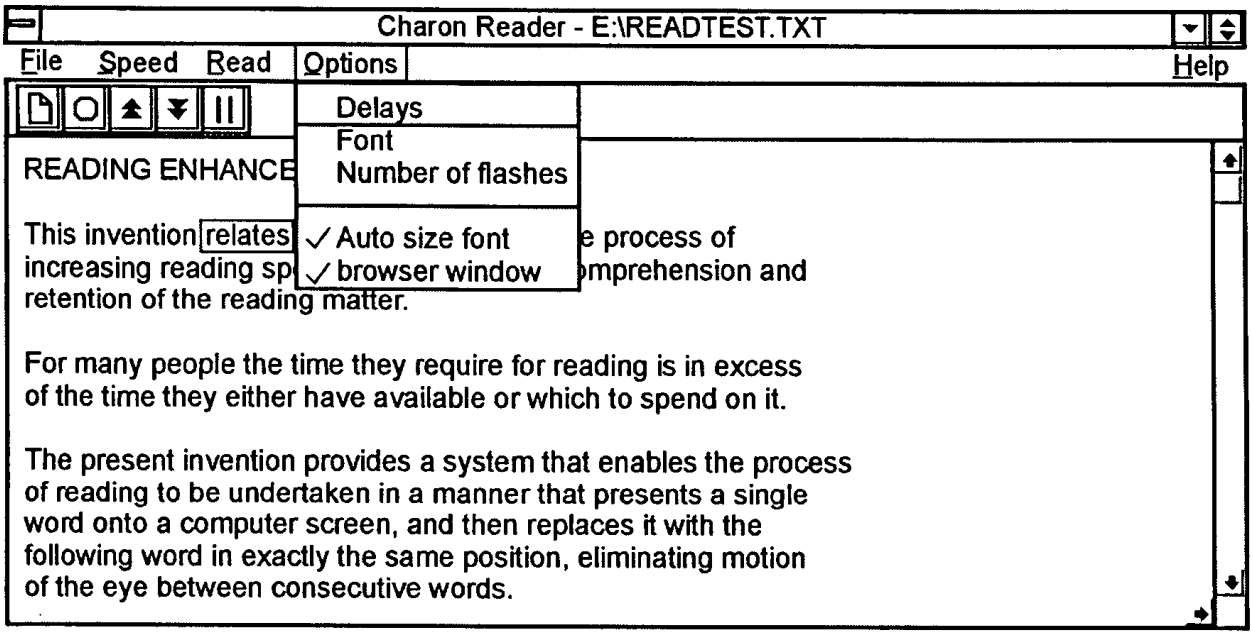

Charon Reader - E:\READTEST.TXT
File Speed Read Options Help
READING ENHANCEMENT PROGRAM
This invention relates to improvements in the process of increasing reading speed and maintaining comprehension and retention of the reading matter.
For many people the time they require for reading is in excess of the time they either have available or which to spend on it.
The present invention provides a system that enables the process of reading to be undertaken in a manner that presents a single word onto a computer screen, and then replaces it with the following word in exactly the same position, eliminating motion of the eye between consecutive words.
Words Per Minute : 0 Interval : 400

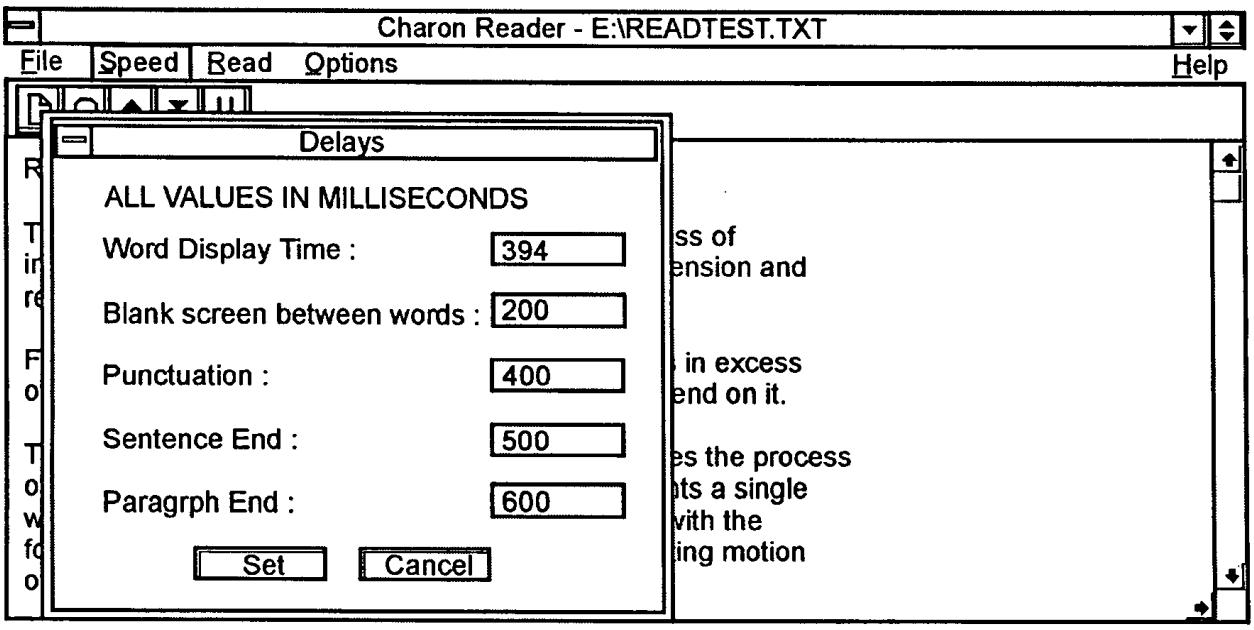
Fig.7.



relates

Define various delay rates

Fig 8


Charon Reader - E:\READTEST.TXT
File
Speed
Read
Options
Help
Delays
ALL VALUES IN MILLISECONDS
Word Display Time :
394
Blank screen between words :
200
Punctuation :
400
Sentence End :
500
Paragrph End :
600
Set
Cancel
ss of
ension and
in excess
end on it.
es the process
ts a single
vith the
ing motion


Words Per Minute : 30 Interval : 394

Fig. 9

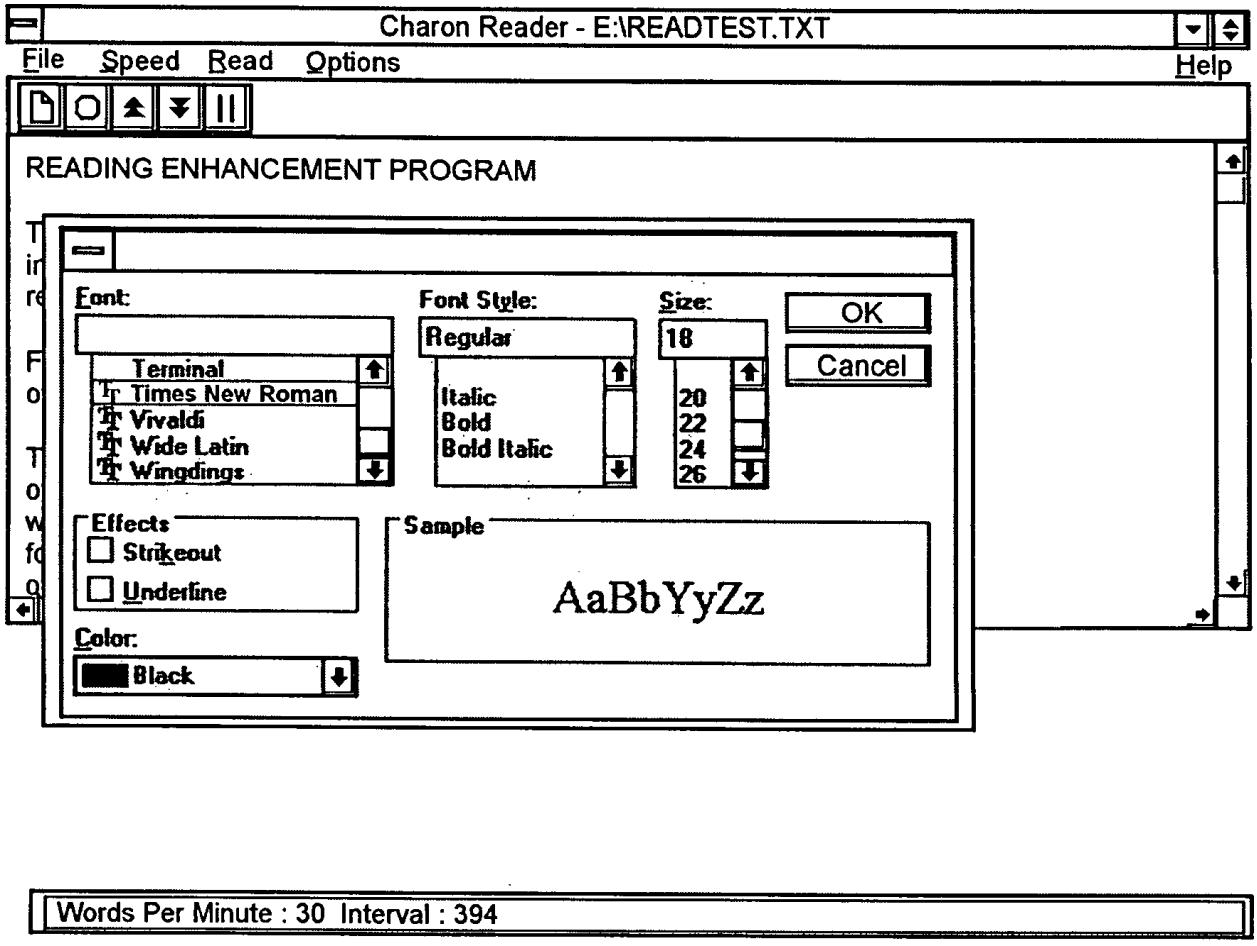

Charon Reader - E:\READTEST.TXT
File
Speed
Read
Options
Help
READING ENHANCEMENT PROGRAM
Font:
Terminal
Times New Roman
Vivaldi
Wide Latin
Wingdings
Font Style:
Regular
Italic
Bold
Bold Italic
Size:
18
20
22
24
26
OK
Cancel
Effects
Strikeout
Underline
Sample
AaBbYyZz
Color:
Black
Words Per Minute : 30 Interval : 394

Fig.10

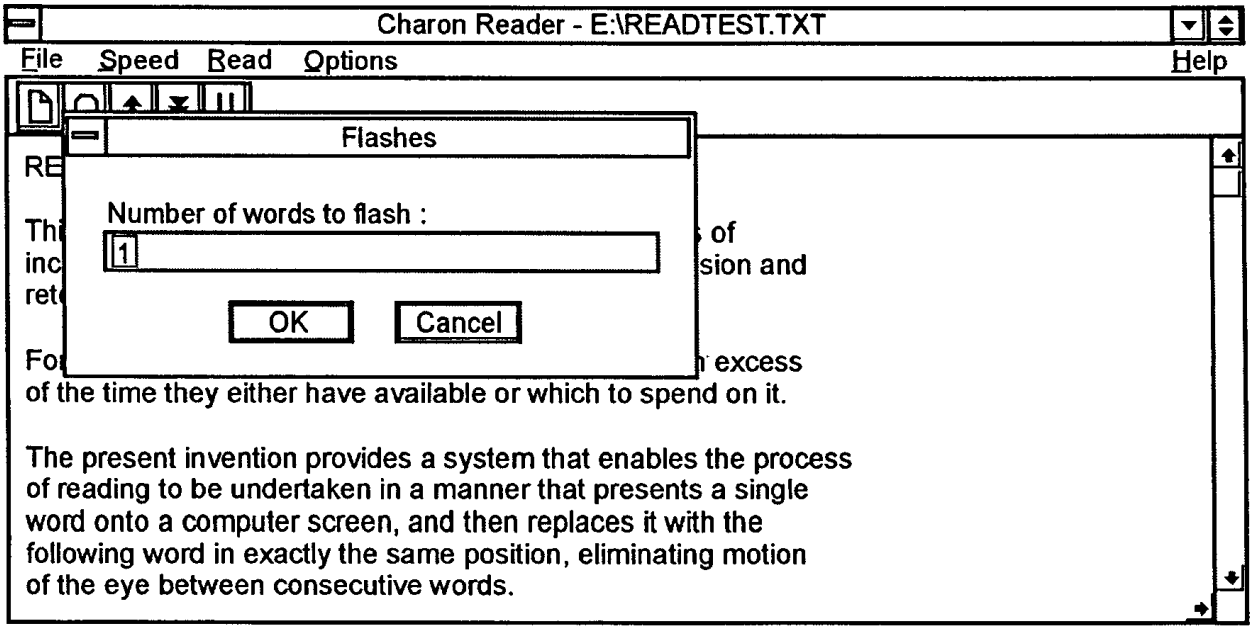
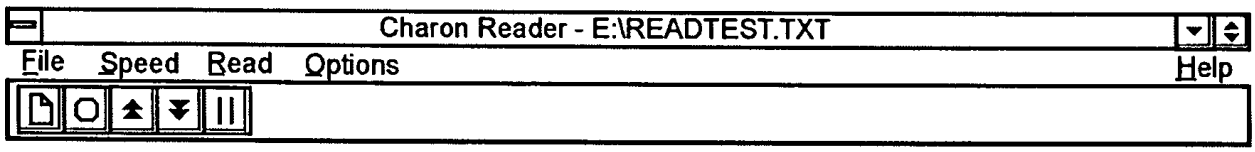

Charon Reader - E:\READTEST.TXT
File
Speed
Read
Options
Help
Flashes
Number of words to flash :
1
OK
Cancel
of
sion and
excess
of the time they either have available or which to spend on it.
The present invention provides a system that enables the process of reading to be undertaken in a manner that presents a single word onto a computer screen, and then replaces it with the following word in exactly the same position, eliminating motion of the eye between consecutive words.
Words Per Minute : 30 Interval : 394

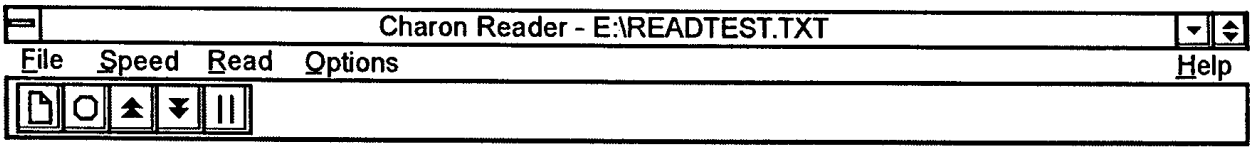
Fig.11


Charon Reader - E:\READTEST.TXT
File
Speed
Read
Options
Help

the

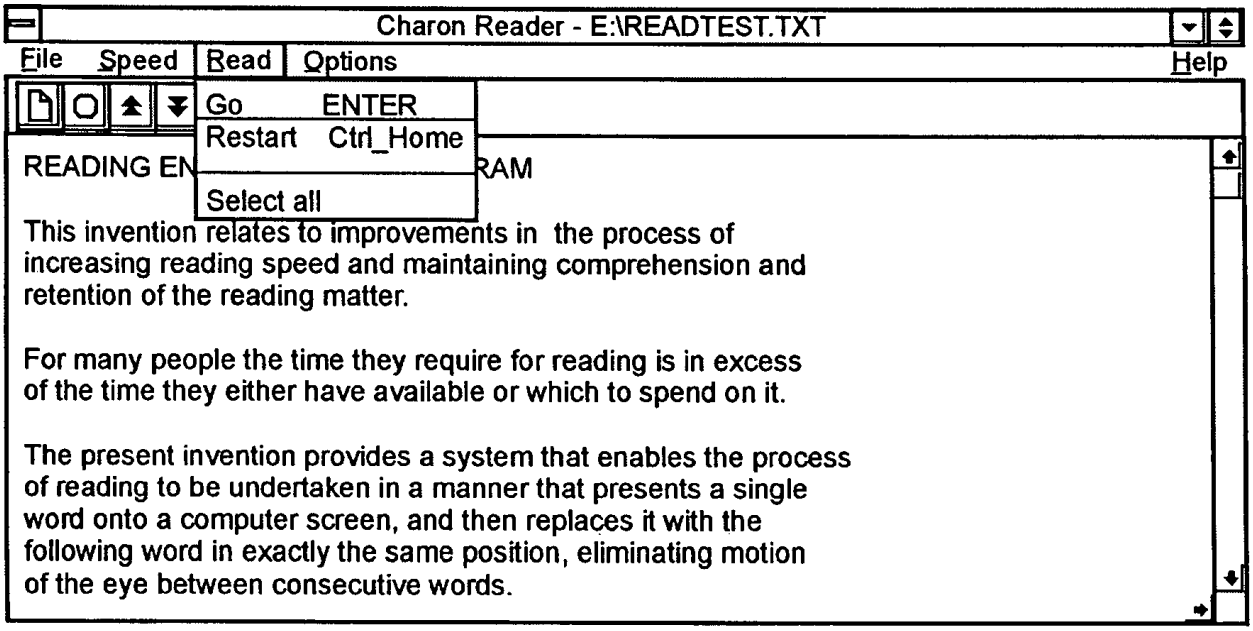

Words Per Minute : 30 Interval : 394

Fig.12



developed

Fig.13


Charon Reader - E:\READTEST.TXT
File
Speed
Read
Options
Help
Go ENTER
Restart Ctrl_Home
Select all
READING EN RAM
This invention relates to improvements in the process of increasing reading speed and maintaining comprehension and retention of the reading matter.
For many people the time they require for reading is in excess of the time they either have available or which to spend on it.
The present invention provides a system that enables the process of reading to be undertaken in a manner that presents a single word onto a computer screen, and then replaces it with the following word in exactly the same position, eliminating motion of the eye between consecutive words.


Begin flashing

Fig.14

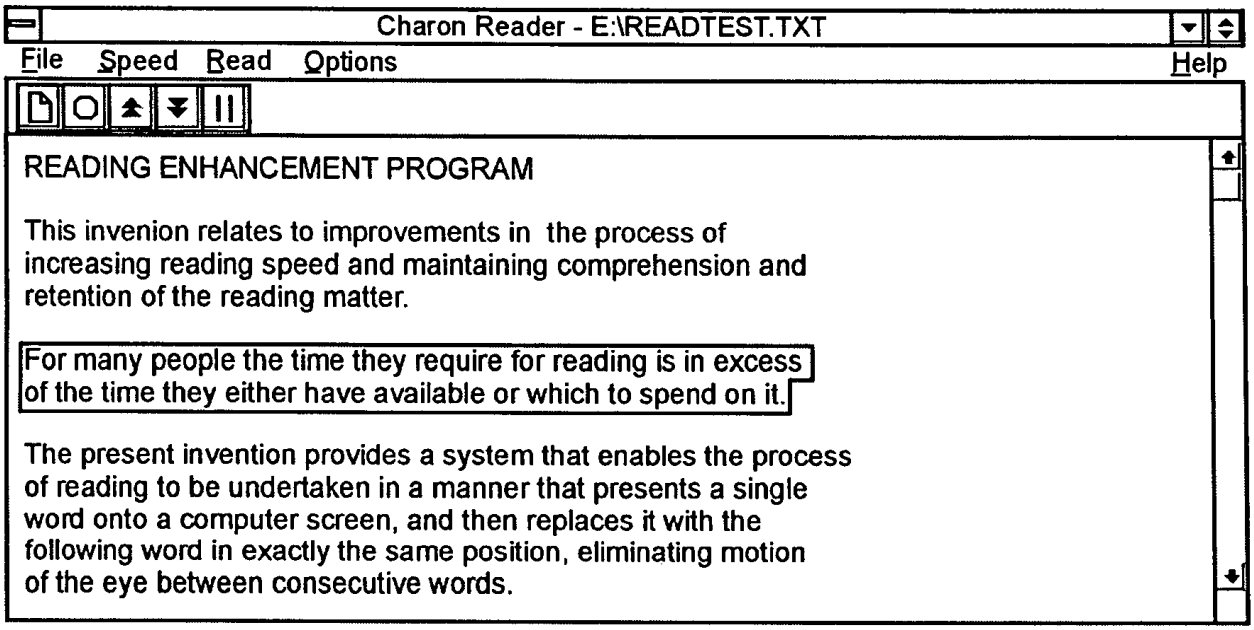
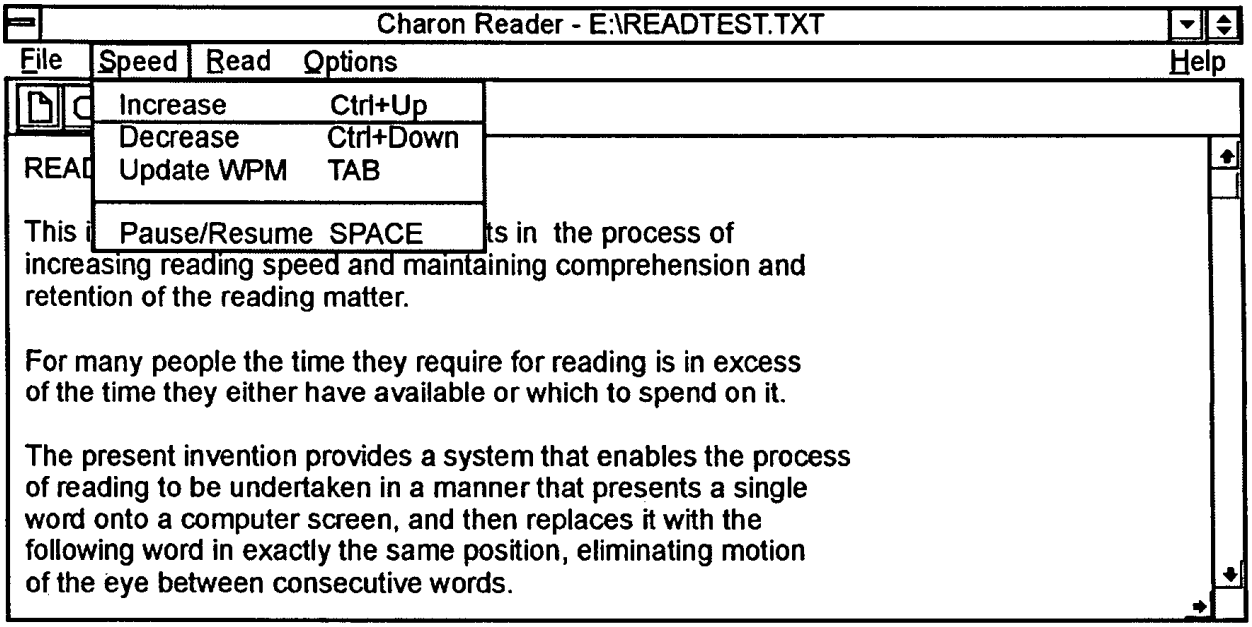
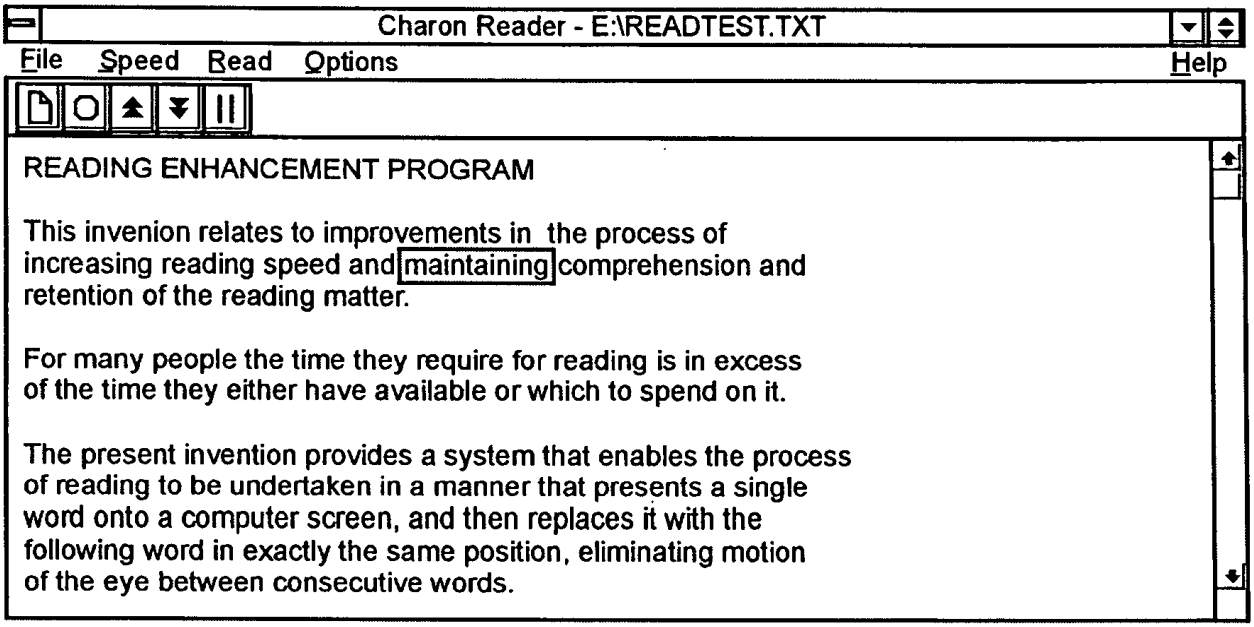

Charon Reader - E:\READTEST.TXT
File
Speed
Read
Options
Help
READING ENHANCEMENT PROGRAM
This invenion relates to improvements in the process of increasing reading speed and maintaining comprehension and retention of the reading matter.
For many people the time they require for reading is in excess of the time they either have available or which to spend on it.
The present invention provides a system that enables the process of reading to be undertaken in a manner that presents a single word onto a computer screen, and then replaces it with the following word in exactly the same position, eliminating motion of the eye between consecutive words.
Words Per Minute : 30 Interval : 394

Fig.15


Charon Reader - E:\READTEST.TXT
File
Speed
Read
Options
Help
Increase Ctrl+Up
Decrease Ctrl+Down
Update WPM TAB
Pause/Resume SPACE
ts in the process of
increasing reading speed and maintaining comprehension and
retention of the reading matter.
For many people the time they require for reading is in excess
of the time they either have available or which to spend on it.
The present invention provides a system that enables the process
of reading to be undertaken in a manner that presents a single
word onto a computer screen, and then replaces it with the
following word in exactly the same position, eliminating motion
of the eye between consecutive words.


Increase rate of flashing

Fig.16.



maintaining

Fig.17

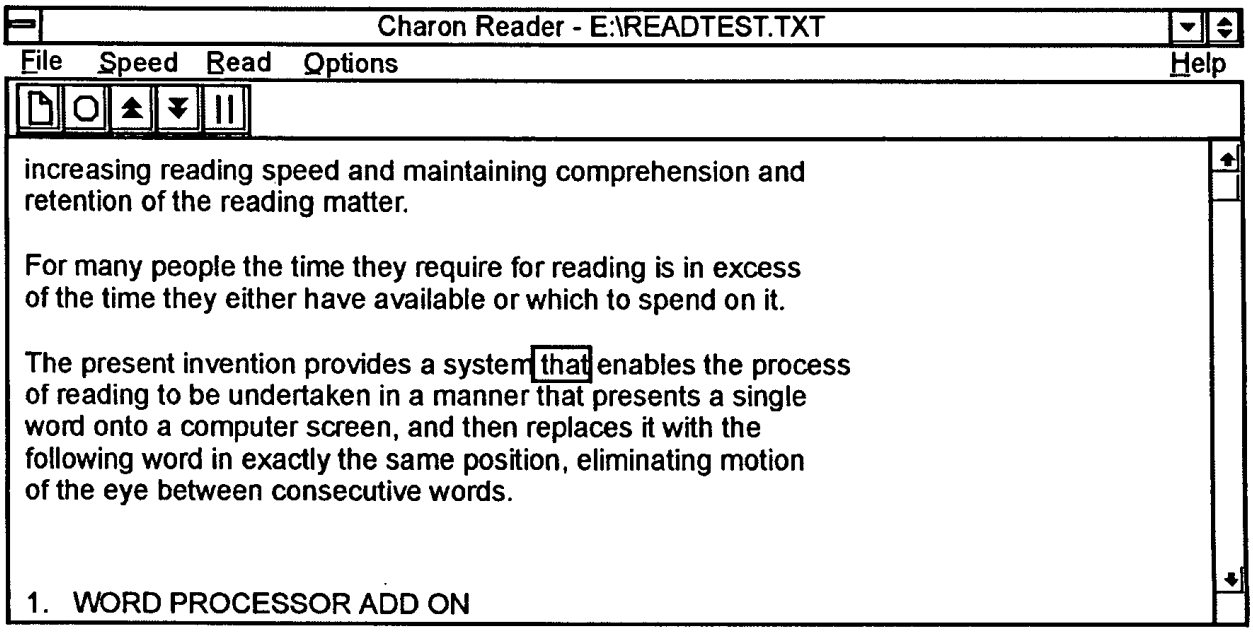

Charon Reader - E:\READTEST.TXT
File Speed Read Options Help
increasing reading speed and maintaining comprehension and retention of the reading matter.
For many people the time they require for reading is in excess of the time they either have available or which to spend on it.
The present invention provides a system that enables the process of reading to be undertaken in a manner that presents a single word onto a computer screen, and then replaces it with the following word in exactly the same position, eliminating motion of the eye between consecutive words.
1. WORD PROCESSOR ADD ON
that
Words Per Minute : 101 Interval : 394

Fig.18