Programmdokumentation zur Anwendung „Animationen“

Entwicklungsumgebung

Die Anwendung wurde auf einem Windows-NT 4.0 System unter der Entwicklungsumgebung Microsoft Visual C++, Version 4.0 erstellt. Die Hardwaremerkmale des Systems sind: Pentium-Prozessor (133 MHz), 64 MByte Hauptspeicher, 4.2 GByte Festplattenspeicher, 21“-Monitor und Matrox-Millenium Grafikkarte (4 MByte).

Programmbeschreibung

Die Anwendung enthält zwei Animationen mit Soundunterstützung. Beide Animationen lassen sich getrennt aufrufen und abspielen. Es handelt sich zum einen um eine Animation, die mit Hilfe der Bibliothek „OpenGL“ realisiert wurde. Sie stellt ein fiktives Universum dar (Abbildung 1), durch das sich der Beobachter bewegen kann. Die Idee dazu entstammt dem Buch „Grafikprogrammierung mit OpenGL“ von Barth, Beier und Pahnke. Realisiert und umgesetzt wurde sie jedoch in Eigenregie nur unter Zuhilfenahme einschlägiger Literatur.

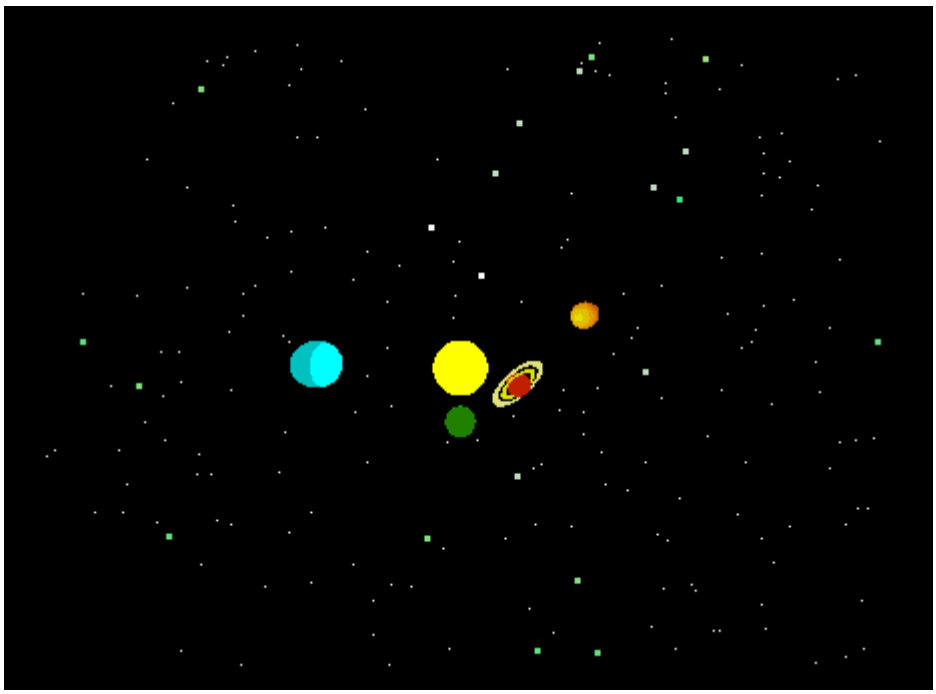

Abbildung 1, Ein Screenshot aus der Universum-Animation mit OpenGL

Die zweite Animation zeigt eine typische Waldwiese (Abbildung 2) auf der sich die verschiedensten Insekten bewegen. Dazu wird eine Melodie ähnlich Vogelgezwitscher abgespielt. Die Idee dazu stammt von mir und wurde durch das Buch „Animationstechniken in Win32“ von Nigel Thompson angeregt. Diese Animation baut vollständig auf der MFC-Bibliothek, Version 4.0 auf.

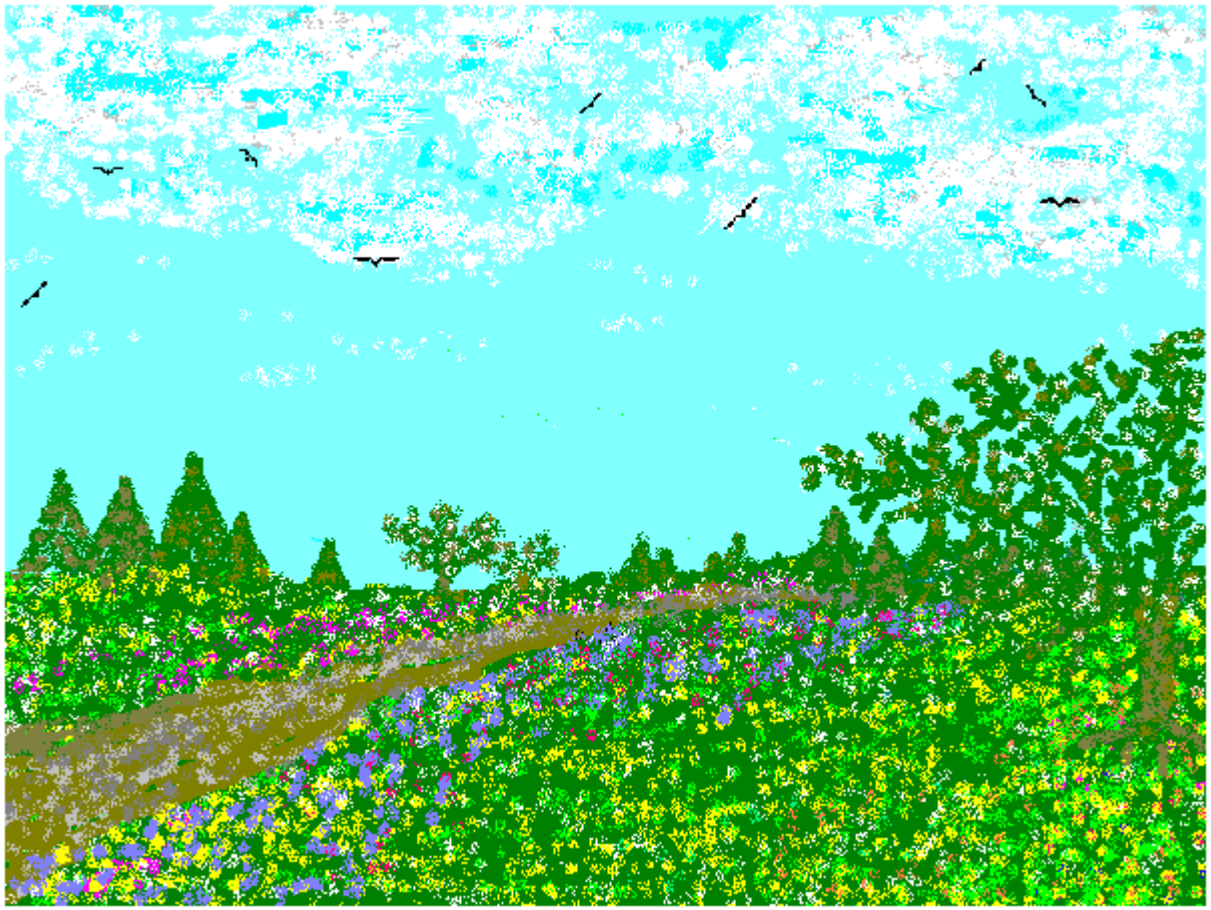

Abbildung 2, Ein Screenshot aus der Waldwiesen-Animation mit MFC

Das Anzeigen und Abspielen der Animationen wird über die Menüleiste und den Animationen zugeordneten Toolboxen realisiert. Jeder Animation ist eine eigene Toolbox zugeordnet, die das Starten und Anhalten dieser ermöglicht. Die Universum-Animation besteht aus zwei getrennt abspielbaren Szenerien und hat deshalb in der ihr gehörigen Toolbox auch doppelt so viele Buttons als die Waldwiesen-Animationstoolbox.

Es folgt anschließend eine Beschreibung der einzelnen Klassen und der in ihnen enthaltenen Elementfunktionen und Datenelemente.

Beschreibung der einzelnen Klassen und der in ihnen enthaltenen Funktionen und Datenelemente (alphabetische Reihenfolge)

CAboutDlg (abgeleitet von CDialog)

Beschreibung:
Stellt einen Informationsdialog zur Anwendung bereit, mit Angaben über Version, Copyright etc..

Datenelemente:
Diese Klasse enthält keine Datenelemente.

Elementfunktionen:
- Konstruktor.

⇒ Zweck: Konstruiert den Dialog mit der zugehörigen Ressourcen-ID.
⇒ Parameter: Keine.

- DoDataExchange. Standardmethode.

⇒ Zweck: Verbindet vorhandene Membervariablen mit ihren zugehörigen Steuerelementen. Da keine existieren, hat die Methode praktisch keine Funktion.

- Parameter: Zeiger auf CDataExchange.

CAnimationenApp (abgeleitet von CWinApp)

Beschreibung:
Das Objekt dieser Klasse realisiert die Erstellung der Anwendung (im Verborgenen), d.h. die Implementierung der Funktion „WinMain“. Mit dem Anlegen eines globalen Objektes von dieser Klasse (theApp) wird dem Programmierer das Realisieren des Anwendungsstarts abgenommen, d.h. er braucht keine Mainfunktion (WinMain) zu schreiben. Diese Funktion wird beim Linken der Anwendung zu den erstellten *.obj - Dateien hinzugefügt. Sie ist in einer hinzugelinkten Bibliothek schon enthalten und übernimmt das Registrieren der Anwendung und den Aufbau der Messageschleife, von der aus die Nachrichten an die einzelnen Klassen weitergeleitet werden. Auch die standardmäßig zu jeder Windowsanwendung gehörenden Funktionen „WindowProc“ und „DefWindowProc“ werden beim Linken hinzugefügt und müssen nicht selbst geschrieben werden. Dabei übernimmt die Funktion „WindowProc“ die Rolle der Hauptfunktion der Anwendung, diese schickt alle Nachrichten an die existierenden Klassenobjekte weiter bzw. nicht von ihr behandelbare Nachrichten an die Funktion „DefWindowProc“.

Datenelemente:
Diese Klasse enthält keine Datenelemente..

Elementfunktionen:
- Konstruktor.

⇒ Zweck: Konstruiert das Anwendungsobjekt.
⇒ Parameter: Keine.

- InitInstance. Standardmethode.

⇒ Zweck: Die Funktion wurde dahingehend erweitert, daß das Hauptfenster nach dem Start maximiert erscheint (ShowWindow, UpdateWindow). Desweiteren wird Titel der Anwendung geändert.

⇒ Parameter: Keine.

⇒ Rückgabe: TRUE bei Erfolg, sonst FALSE.

- OnAppAbout. Standardmethode (Ereignisbehandler).

⇒ Zweck: Zeigt einen modalen Dialog (Anwendungsinfo) an.

- Parameter: Keine.

CAnimationenDoc (abgeleitet von CDocument)

Beschreibung:
Diese Klasse bzw. das Objekt von ihr stellt die Daten der Anwendung bereit. In diesem Fall existieren keine Daten im eigentlichen Sinne. Aus diesem Grund ist der Mechanismus des Document-Objektes in dieser Anwendung überflüssig und es wurden keine Veränderungen an ihm (Objekt) vorgenommen, d.h. es wurde so belassen, wie es vom AppWizard erzeugt wurde.

Datenelemente:
Diese Klasse enthält keine Datenelemente.

Elementfunktionen:

- AssertValid. Standardmethode, wurde nicht geändert.

⇒ Zweck: Abbruch des Prozeßstarts, wenn erzeugen des Documents fehlschlug.

⇒ Parameter: Keine.

- Konstruktor.

⇒ Zweck: Keine Funktion.

⇒ Parameter: Keine.

- Destruktor.

⇒ Zweck: Keine Funktion.

- Dump. Standardmethode.

⇒ Zweck: Zum debuggen in der Debug-Version. Keine Funktion in der Release. Ruft hier die Standardmethode Dump() von CDocument auf.

⇒ Parameter: Debugkontextbezogene Variable.

- OnNewDocument.

⇒ Zweck: Ruft als erstes die Basisfunktion (OnNewDocument()) der Klasse auf. Hat aber ansonsten keinen Zweck, wie der Beschreibung zur Klasse zu entnehmen ist.

⇒ Parameter: Keine.

- Serialize. Standardmethode.

⇒ Zweck: Stellt einen Mechanismus zum Öffnen und Speichern von Dokumenten bereit, hat aber in diesem Fall keinen Zweck, wie der Beschreibung zur Klasse zu entnehmen ist.

⇒ Parameter: Die Adresse eines CArchive-Objektes. Diese wird bei Benutzung des bereitgestellten Mechanismus zum Speichern bzw. Öffnen von Dateien verwendet.

CAnimationenView (abgeleitet von CView)

Beschreibung:
Fungiert als Hauptklasse der Anwendung, da sie das Darstellen und Abspielen der Animationen ermöglicht. Mit Hilfe ihrer Elementfunktionen ist es möglich, grafische Ausgaben zu realisieren, d.h. zum Beispiel Farben darzustellen oder Bitmaps als Hintergrund anzuzeigen oder auch kleine Sprites (kleine Bitmaps, die sich bewegen können) auf den Bildschirm zu bringen. Aus dieser Klasse heraus werden die beiden Animationen gesteuert. Zu diesem Zweck übernehmen Hilfsfunktionen dieser Klasse auch das Bearbeiten von Befehlen aus der Menüleiste und den Animationen zugeordneten Toolboxen.

Datenelemente:
- m_pDC (Typ CClientDC*, private). Stellt einen Zeiger auf den Devicekontext der Sicht bereit und wird an das Objekt der OpenGL- und der Visual-C++-Animation übergeben, damit deren Funktionen die grafische Darstellung realisieren können.
- m_pGLAni (Typ COpenGLUniverse* , private). Stellt einen Zeiger auf das Objekt der OpenGL-Animation dar. Damit können die Funktionen der Sicht die Elementfunktionen der Klasse zur OpenGL-Animation aufrufen und auch auf öffentliche Datenelemente dieser zugreifen.
- m_iViewSizeX (Typ int, private). Die Ausdehnung der Sicht in X-Richtung.
- m_iViewSizeY (Typ int, private). Die Ausdehnung der Sicht in Y-Richtung. Anhand der beiden Ausdehnungsvariablen wird der Viewport für die OpenGL-Animation eingestellt.
- m_uiWelcherGLButton1 (Typ UINT, private). Speichert die ID des Buttons der gedrückt wurde.
- m_uiWelcherGLButton2 (Typ UINT, private). Speichert die ID des Buttons der gedrückt wurde.
- m_iOpenGLBeobachterposition (Typ int, private). Die Position des Beobachterstandpunktes innerhalb der Universum-Animation. Damit läßt sich Teil 1 dieser Animation aus der Sicht heraus steuern.
- m_bGLAniPlaneten (Typ BOOL, private). Zeigt an, ob der Teil der Universum-Animation, der die Planeten betrifft, abgespielt werden soll.
- m_bGLAniBeobachter (Typ BOOL, private). Zeigt an, ob der Teil der Universum-Animation, der den Beobachter betrifft, abgespielt werden soll.
- m_pVCAni (Typ CGrueneWiese*, private). Stellt einen Zeiger auf das Objekt der Visual-C++-Animation dar. Damit können die Funktionen der Sicht die Elementfunktionen der Klasse zur Visual-C++-Animation aufrufen und auch auf öffentliche Datenelemente dieser zugreifen.
- m_uiWelcherVCButton (Typ UINT, private). Speichert die ID des Buttons der gedrückt wurde.
- m_bIsOpenGL (Typ BOOL, private). Zeigt an, daß die GL-Demo abgespielt wird.
- m_bIsCPlusDemo (Typ BOOL, private). Zeigt an, daß die C++-Demo abgespielt wird.
- m_uiTimer (Typ UINT, private). Für den Start des Timer bzw. dessen löschen.

Elementfunktionen:
- AssertValid. Standardmethode, wurde nicht geändert.

⇒ Zweck: Abbruch des Prozeßstarts, wenn erzeugen der Sicht fehlschlug. Ruft die Basisfunktion der Klasse auf.

⇒ Parameter: Keine.

- Konstruktor.

⇒ Zweck: Konstruiert das Sichtobjekt und initialisiert die Zeiger auf die zwei Animationsobjekte. Desweiteren werden die beiden Flags, die anzeigen, ob und welche Animation gerade aktiv ist, auf FALSE gesetzt, d.h. zu Programmstart ist keine Animation aktiv.

⇒ Parameter: Keine.

- Destruktor.

⇒ Zweck: Die Zeiger auf die beiden Animationsobjekte werden wieder freigegeben und eventuell noch abgespielter Sound wird gestoppt. Dies wird durch den Aufruf der Funktion „sndPlaySound“ mit NULL als erstem Parameter realisiert.

- Dump. Standardmethode.

⇒ Zweck: Zum debuggen in der Debug-Version. Keine Funktion in der Release. Ruft hier die Standardmethode Dump() von CView auf.

⇒ Parameter: Debugkontextbezogene Variable.

- GetDocument. Standardmethode.

⇒ Zweck: Liefert als Rückgabewert einen Zeiger auf das Documentobjekt, welches der Sicht zugeordnet ist. Mit diesem Rückgabewert initialisiere ich einen modulglobalen Zeiger, welcher dann über die gesamte Laufzeit allen Sichtfunktionen schnell zur Verfügung steht.

⇒ Parameter: Keine.

⇒ Rückgabe: Zeiger auf das Documentobjekt.

- OnCplusDemo. Menübefehlbehandler.

⇒ Zweck: Togglet das Animations-Aktiv-Flag. In Abhängigkeit von dessen Zustand wird entweder die erste Szene der Animation angezeigt (Flag ist auf TRUE) und die zugeordnete Toolbox zum Starten und Anhalten der Szenerie eingeblendet oder die Animation und die Toolbox werden ausgeblendet (Flag ist auf FALSE). Das Ein- und Ausblenden der Toolbox geschieht mit Hilfe der Funktionen „ShowWindow“ und „RecalcLayout“. Beides sind Methoden des CMainFrame-Objektes und aus diesem Grund muß ein Zeiger auf dieses Objekt initialisiert werden (CMainFrame* pFrame = (CMainFrame*) AfxGetApp()->m_pMainWnd).

⇒ Parameter: Keine.

- OnCreate. Standardmethode.

⇒ Zweck: Zuerst wird die Basisfunktion der Klasse aufgerufen. Gibt diese einen Erfolgswert zurück, wird anschließend ein Zeiger auf die Sicht initialisiert. Dieser wird den beiden Animationsobjekten über ihre Create()-Funktion übergeben. Damit sind diese in der Lage grafische Ausgaben auf dem Bildschirm sichtbar zu produzieren. Der Zeiger stellt einen Devicekontext dar.

⇒ Parameter: Struktur vom Typ LPCREATESTRUCT. Damit lassen sich Anfangswerte der Sicht manipulieren. Dazu verändert man die Struktur, bevor sie an die Basisfunktion der Sicht weitergereicht wird. In diesem Fall bleibt sie unverändert.

⇒ Rückgabe: -1 bei Mißerfolg, sonst 0.

- OnDraw. Standardmethode.

⇒ Zweck: Hauptfunktion der gesamten Anwendung. Sie wird jedesmal aufgerufen, wenn das Fenster neu gezeichnet werden muß, d.h. in ihr müssen alle Anweisungen stehen, die gewährleisten, daß nach dem Neuzeichnen des Fensters der Zustand wie zuvor dargestellt wird. Die eigentliche Hauptfunktion zum Neuzeichnen ist OnPaint(). Diese wird aber verborgen. Sie schickt jedoch die Nachricht WM_PAINT und dadurch wird der Aufruf von OnDraw() realisiert. Die Funktion OnDraw() in dieser Anwendung übernimmt nur eine Mittlerrolle. Trifft die Nachricht WM_PAINT ein, ruft sie entweder die Methode „Animation“ der beiden Animationsobjekte auf, wenn gerade eine Animation aktiv ist oder sie zeichnet den Hintergrund des Anwendungsfensters schwarz.

⇒ Parameter: Devicekontext, welcher einen Zeiger auf die Sicht repräsentiert.

- OnGLPalette1. Toolboxbefehlbehandler.

⇒ Zweck: Startet oder stoppt den Teil der Universum-Animation, der die Planeten betrifft. Dazu wird ein entsprechendes Flag gesetzt, welches von der Timerfunktion ausgewertet wird.

⇒ Parameter: Die ID des Buttons.

- OnGLPalette2. Toolboxbefehlbehandler.

⇒ Zweck: Startet oder stoppt den Teil der Universum-Animation, der den Beobachterstandpunkt betrifft. Dazu wird ein entsprechendes Flag gesetzt, welches von der Timerfunktion ausgewertet wird. Weiterhin wird das Abspielen einer Hintergrundmelodie gestartet oder angehalten.

⇒ Parameter: Die ID des Buttons.

- OnOpenglDemo. Menübefehlbehandler.

⇒ Zweck: Togglet das Animations-Aktiv-Flag. In Abhängigkeit von dessen Zustand wird entweder die erste Szene der Animation angezeigt (Flag ist auf TRUE) und die zugeordnete Toolbox zum Starten und Anhalten der Szenerie eingeblendet oder die Animation und die Toolbox werden ausgeblendet (Flag ist auf FALSE). Weiterhin werden für die Animation Starteinstellungen vorgenommen (Beobachterstandpunkt und Stellung der Planeten) und der OpenGL-Support initialisiert. Beim Ausblenden der Szenerie wird auch ein eventuell noch spielender Sound angehalten.

⇒ Parameter: Keine.

- OnSize. Standardmethode.

⇒ Zweck: Ruft als erstes die Basisfunktion der Klasse auf und speichert anschließend die X- und Y-Ausdehnung der Sicht. Wichtig ist dies beim Anzeigen der OpenGL-Animation, da diese nur einen kleinen Ausschnitt der Sicht in Anspruch nimmt (aus Performancegründen) und dieser in der Mitte dargestellt werden soll.

⇒ Parameter: Die Art der Größenänderung (z.B. Maximierung) und die Werte für X- und Y-Ausdehnung.

- OnTimer. Standardmethode.

⇒ Zweck: Verändert die Animationsparameter (Einstellungen des nächsten Frames) und ruft dann die Methode „Animation“ der beiden Animationsobjekte auf, in Abhängigkeit davon, welche Animation gerade angezeigt wird. Nachdem alle Anweisungen abgearbeitet sind, wird die Basisfunktion der Klasse aufgerufen.

⇒ Parameter: Die ID des Timerereignisses. Dazu werden zwei DEFINES benutzt, welche die zur Zeit aktive Animation identifizieren.

- OnUpdateCplusDemo. Menübefehlbehandler.

⇒ Zweck: Aktiviert bzw. deaktiviert den entsprechenden Menübefehl und zeigt ein Häkchen daneben an, wenn die dazugehörige Animation gerade angezeigt wird.

⇒ Parameter: Zeiger auf ein Objekt vom Typ CCmdUI. Dieses ist dem Menübefehl zugeordnet.

- OnUpdateGLPalette1. Toolboxbefehlbehandler.

⇒ Zweck: Aktiviert bzw. deaktiviert den entsprechenden Toolboxbutton.

⇒ Parameter: Zeiger auf ein Objekt vom Typ CCmdUI. Dieses ist dem Toolboxbefehl zugeordnet.

- OnUpdateGLPalette2. Toolboxbefehlbehandler.

⇒ Zweck: Aktiviert bzw. deaktiviert den entsprechenden Toolboxbutton.

⇒ Parameter: Zeiger auf ein Objekt vom Typ CCmdUI. Dieses ist dem Toolboxbefehl zugeordnet.

- OnUpdateOpenglDemo. Menübefehlbehandler.

⇒ Zweck: Aktiviert bzw. deaktiviert den entsprechenden Menübefehl und zeigt ein Häkchen daneben an, wenn die dazugehörige Animation gerade angezeigt wird.
⇒ Parameter: Zeiger auf ein Objekt vom Typ CCmdUI. Dieses ist dem Menübefehl zugeordnet.

- OnUpdateVCPalette. Toolboxbefehlbehandler.

⇒ Zweck: Aktiviert bzw. deaktiviert den entsprechenden Toolboxbutton.
⇒ Parameter: Zeiger auf ein Objekt vom Typ CCmdUI. Dieses ist dem Toolboxbefehl zugeordnet.

- OnVCPalette. Toolboxbefehlbehandler.

⇒ Zweck: Startet oder stoppt die Waldwiesen-Animation. Dazu wird ein entsprechendes Flag gesetzt, welches von der Timerfunktion ausgewertet wird. Weiterhin wird das Abspielen einer Hintergrundmelodie gestartet oder angehalten.
⇒ Parameter: Die ID des Buttons.

- PreCreateWindow. Standardmethode.

⇒ Zweck: Die als Parameter übergebene Struktur vom Typ CREATESTRUCT (bzw. deren Adresse), muß dahingehend geändert werden, daß der OpenGL-Support unterstützt wird. Dazu muß der Teil, der den Stil der Sicht betrifft (style) auf die Werte WS_CLIPSIBLINGS und WS_CLIPCHILDREN gesetzt werden. Mit dieser modifizierten Struktur wird dann die Basisfunktion der Klasse aufgerufen.
⇒ Parameter: Adresse einer Struktur vom Typ CREATESTRUCT.

CGrueneWiese (keine Basisklasse)

Beschreibung:
Das Objekt dieser Klasse stellt Datenelemente und Funktionen zur Verfügung, damit die implementierte Visual-C++-Animation dargestellt werden kann. Die Animation stellt eine Waldwiesenszenerie dar, auf der sich die verschiedensten Insekten bewegen. Dazu wird eine Hintergrundmelodie abgespielt. Zu Beginn der Animation befindet sich die Szene in Ruhe. Durch entsprechende Schaltflächen einer Palette (diese wurde vom Objekt CMainFrame erstellt und versteckt und wird vom Objekt CVektorgrafikenView aktiviert d.h. angezeigt) kann die Animation gestartet und angehalten werden.
Die Klasse benötigt keine Basisklasse, da sie den einzig wichtigen Parameter (ein Zeiger auf den der Sicht zugeordneten Devicekontext) von der Sicht gestellt bekommt. Damit ist es den Elementfunktionen dieser Klasse möglich, die grafische Ausgabe zu realisieren, die nur mit Hilfe von Win32-Funktionen vorgenommen wird.
Die Idee zu der Szenerie stammt von mir und wurde durch das Buch „Animationstechniken in Win32“ von Nigel Thompson inspiriert. Umgesetzt wurde sie mit Hilfe einer von diesem Autor bereitgestellten Library.

Datenelemente:

- m_bmpHintergrund (Typ CBitmap, private). Nimmt die Hintergrundbitmap der Animation auf.
- m_rectWindow (Typ RECT, private). Speichert die aktuelle Größe der Sicht.
- m_pDC (Typ CClientDC*, protected). Devicekontextzeiger um die Animation auf den Bildschirm bringen zu können.

Elementfunktionen:

- Animation.

⇒ Zweck: Anzeigen des aktuellen Bildes der Animation. Der Teil des Fensters,

der nicht von dem Hintergrundbild bedeckt wird, wird schwarz dargestellt.

⇒ Parameter: Keine.

- Konstruktor.

⇒ Zweck: Zur Zeit noch keine Funktion.

⇒ Parameter: Keine.

- Destruktor.

⇒ Zweck: Zur Zeit noch keine Funktion.

- Create.

⇒ Zweck: Initialisiert den modulglobalen Devicekontextzeiger mit dem von der Sicht übergebenen. Dieser Zeiger ist vom Typ CClientDC und gibt somit den Elementfunktionen dieser Klasse ein Handle zur grafischen Ausgabe.

⇒ Parameter: Zeiger auf die Sicht bzw. Devicekontext.

- InitialisiereAnimation.

⇒ Zweck: Lädt das Hintergrundbild in Form eines CBitmapobjektes und speichert die Größe der Sicht.

⇒ Parameter: Größe der Sicht in Form einer RECT-Struktur.

CMainFrame (abgeleitet von CFrameWnd)

Beschreibung:
Mit Hilfe der in dieser Klasse enthaltenen Elementfunktionen wird der Hauptrahmen der Anwendung erstellt. Das bedeutet einstellen des Aussehens der Anwendung nach dem Start, erstellen von Symbolleisten, Statuszeile, Menüs (verborgen im Hintergrund, durch den AppWizard) etc. und einrichten von zur Laufzeit benötigten Objekten (z.B. zu Anzeigefenstern umfunktionierte Editierfelder). Die Arbeiten die hier geleistet werden, betreffen zum größten Teil die Schnittstelle Mensch ←→ Anwendung.

Datenelemente:

- m_wndOpenGLBox (Typ CToolBar, public). Objekt zur Aufnahme der Toolbox-Palette, welche der OpenGL-Animation zugeordnet ist.
- m_wndVCBox (Typ CToolBar, public). Objekt zur Aufnahme der Toolbox-Palette, welche der Visual-C++-Animation zugeordnet ist.

Beide Paletten sind „öffentlich“, um der Sicht bzw. ihren Elementfunktionen einen Zugriff darauf gewähren zu können.

Elementfunktionen:

- AssertValid. Standardmethode.

⇒ Zweck: Abbruch des Prozeßstarts, wenn erzeugen des MainFrame fehlschlug. Es wird die Basisfunktion der Klasse aufgerufen.

⇒ Parameter: Keine.

- Konstruktor.

⇒ Zweck: Zur Zeit noch keine Funktion.

⇒ Parameter: Keine.

- Destruktor.

⇒ Zweck: Zur Zeit noch keine Funktion.

- Dump. Standardmethode.

⇒ Zweck: Zum debuggen in der Debug-Version. Keine Funktion in der Release. Ruft hier die Standardmethode Dump() von CFrameWnd auf.

⇒ Parameter: Debugkontextbezogene Variable.

- OnCreate. Standardmethode.

⇒ Zweck: Aufruf der Basismethode OnCreate() von CFrameWnd mit erhaltenem Parameter. Danach werden die den Animationen zugeordneten Toolbox-Paletten erstellt (aber nicht angezeigt). Erstellen der nicht sichtbaren Palette für das Steuern der OpenGL-Animation. Die Palette hat vier Schaltflächen, unterteilt in zwei Gruppen (die Animation besteht aus zwei Teilen und jeder Teil hat seine eigenen Steuerschaltflächen). Die Größe und der Stil der Buttons werden geändert. Erstellen der nicht sichtbaren Palette für das Steuern der Visual-C++-Animation. Diese Palette hat zwei Schaltflächen, deren Größe ebenfalls geändert wird.

⇒ Parameter: Zeiger auf eine Struktur vom Typ CREATESTRUCT. Mit diesem Parameter muß als erstes die Basismethode OnCreate() von CFrameWnd aufgerufen werden.

⇒ Rückgabe: Wert 0 bei vollständigem Erfolg sonst -1.

- OnGetMinMaxInfo. Standardmethode.

⇒ Zweck: Diese Funktion stellt die Mindestgröße des Hauptfensters (875x655 Pixel) ein (verändern der Struktur vom Typ MINMAXINFO). Dies wird durch Aufruf der Basisfunktion von CFrameWnd mit geänderter Struktur realisiert.

⇒ Parameter: Far-Pointer auf eine Struktur vom Typ MINMAXINFO.

- PreCreateWindow. Standardmethode.

⇒ Zweck: Hier wird die Struktur vom Typ CREATESTRUCT so geändert, daß das Fenster in der linken oberen Ecke mit einer gewissen (von der Bildschirmauflösung abhängigen) Größe dargestellt wird, wenn es nicht im maximierten oder minimierten Zustand ist d.h., wenn es den Normalzustand hat. Durch aufrufen der Basisfunktion von CFrameWnd (mit der geänderten Struktur) wird dies realisiert. Desweiteren wird der Stil des Fensters (style, dwExStyle) so geändert, daß es keinen MIN-MAX-Button besitzt und immer über allen anderen Fenstern steht (WS_EX_TOPMOST).

⇒ Parameter: Adresse der Struktur vom Typ CREATESTRUCT.

⇒ Rückgabe: Rückgabewert der Basisfunktion (TRUE oder FALSE).

COpenGLUniverse (keine Basisklasse)

Beschreibung:
Das Objekt dieser Klasse stellt Datenelemente und Funktionen zur Verfügung, damit die implementierte OpenGL-Animation dargestellt werden kann. Die Animation stellt ein Universum dar (mit zur Zeit vier Planeten und einer Sonne). Zu Beginn der Animation befinden sich die Planeten und der Beobachter in Ruhe. Durch entsprechende Schaltflächen einer Palette (diese wurde vom Objekt CMainFrame erstellt und versteckt und wird vom Objekt CVektorgrafikenView aktiviert, d.h. angezeigt) kann die Planetenrotation um die Sonne gestartet und angehalten und der Beobachterpunkt durch den Raum verlagert und die Bewegung dessen ebenfalls angehalten werden.
Die Klasse benötigt keine Basisklasse, da sie vollständig auf der OpenGL aufsetzt und keine Kommunikation mit dem Benutzer hat.
Die Idee zu der Szenerie stammt aus dem Buch „Grafikprogrammierung mit OpenGL“ von Barth, Beier und Pahnke. Realisiert und umgesetzt wurde sie jedoch in Eigenregie nur unter Zuhilfenahme einschlägiger Literatur.
Die OpenGL eröffnet völlig neue Wege in der Grafik- und Animationsprogrammierung unter Windows-NT, da sie es dem Programmierer durch benutzen von einfach zu handhabenden Funktionen gestattet, komplizierte mathematische Modelle und Theorien (die Grundlage der Grafischen DV sind) umzusetzen, ohne sich um die Realisierung dieser kümmern zu müssen. Genannt sei hier als Beispiel die Beleuchtungstheorie von 3-dimensionalen Körpern.

Datenelemente:
- m_iSterne (Typ int, private). Displaylistenidentifikator.
- m_iSonne (Typ int, private). Displaylistenidentifikator.
- m_iMilchstrasse (Typ int, private). Displaylistenidentifikator.
- m_iAnimation1 (Typ int, private). Displaylistenidentifikator.
- m_iAnimation2 (Typ int, private). Displaylistenidentifikator.
- m_iP1Winkelanzahl (Typ int, public). Die Anzahl der Winkel, in die die Rotationsebene des Planeten unterteilt ist. Je größer sie ist, desto geschmeidiger verläuft die Rotation.
- m_iP2Winkelanzahl → siehe zuvor.
- m_iP3Winkelanzahl → siehe zuvor.
- m_iP4Winkelanzahl → siehe zuvor.
- m_fP1Winkel (Typ float, public). Die Größe der Winkel, in die die Rotationsebene des Planeten unterteilt ist. Sie korrespondiert mit der Anzahl der Winkel. Je kleiner die Größe ist, desto geschmeidiger verläuft die Rotation.
- m_fP2Winkel → siehe zuvor.
- m_fP3Winkel → siehe zuvor.
- m_fP4Winkel → siehe zuvor.
- m_fvpz (Typ float, public). Veränderbare Tiefeninformationen für den Augpunkt.
- m_frpz (Typ float, public). Veränderbare Tiefeninformationen für den Blickpunkt.
- m_fvpx (Typ float, public). Veränderbare X-Koordinaten des Augpunktes (zum Drehen auf der Stelle).
- m_frpx (Typ float, public). Veränderbare X-Koordinaten des Blickpunktes (zum Drehen auf der Stelle).
- m_pDC (Typ CClientDC*, protected). Devicekontextzeiger um den OpenGL-Support initialisieren und die Szenen ausgeben zu können.

Alle Public-Variablen sind deswegen öffentlich, um aus der Sicht heraus einen Zugriff darauf zu haben, damit von dort über den entsprechenden Timer die Animation gesteuert werden kann. Dies widerspricht zwar einer sauberen objektorientierten Lösung, geht dafür aber wesentlich schneller.

Elementfunktionen:

- Animation.

⇒ Zweck: Anzeigen des aktuellen Bildes und indirekt auch Berechnen des nächsten. Es werden die zwei Animationsdisplaylisten aufgerufen, der Aug- und Blickpunkt eingestellt und die Position der Planeten dargestellt. Die GL-Funktion SwapBuffers() angewandt auf den aktuellen DC bringt das Bild zur Anzeige (ruft automatisch glFinish() auf).

⇒ Parameter: Keine.

- Animationslisten.

⇒ Zweck: Baut die zwei Displaylisten für die Animation auf. Displaylisten können unter Umständen Geschwindigkeitsgewinne bringen, sind aber entsprechend des ARB (Architectural Review Board) nicht langsamer als ein direkter Aufruf der in den Listen enthaltenen Befehle. Geschwindigkeitsgewinne ergeben sich jedoch nur, wenn die OpenGL auf einem Netzserver liegt. In diesem Fall werden alle innerhalb einer Liste befindlichen Anweisungen auf dem Server gehalten und nur der CallList()-Aufruf geht über das Netz. Möglich ist dies mit allen Anweisungen, die keinen direkten Einfluß auf die Animation (falls es sich um eine solche handelt) haben.

⇒ Parameter: Keine.

- Konstruktor.

⇒ Zweck: Initialisiert die Displaylistenidentifikatoren und die Augpunktkoordinaten.

⇒ Parameter: Keine.

- Destruktor.

⇒ Zweck: Hat zur Zeit keine Funktion.

- Create.

⇒ Zweck: Initialisiert den Zeiger auf die Sicht mit dem übergebenen Zeiger.

⇒ Parameter: Zeiger auf die Sicht.

- InitialisiereAnimation.

⇒ Zweck: Nimmt Grundeinstellungen der GL vor. Einstellen der Beleuchtungssituation, Aktivierung des Z-Buffer-Tests, Deaktivierung des Zeichnens der Objektrückseiten, Funktionsaufrufe zur Szenerieeinstellung und Displaylistenerzeugung.

⇒ Paramter: Keine.

- InitialisiereGL.

⇒ Zweck: Diese Funktion (in Zusammenspiel mit SetzePixelFormat()) ist Grundvoraussetzung damit die OpenGL-Befehle auch sichtbar wirksam werden, d.h. damit im Endeffekt das Bild, welches durch GL-Funktionen aufgebaut wurde, auch angezeigt werden kann. Durch Aufruf von SetzePixelFormat() (siehe auch Beschreibung dazu) wird der GL-Support initialisiert und danach wird ein Kontext für die GL-Funktionen erstellt und realisiert. Die Funktionen zur Gerätekontextbereitstellung kommen aus der Windows-NT-Implementierung der OpenGL (aus der Library WGL).

⇒ Parameter: Keine.

⇒ Rückgabe: Bei Erfolg TRUE ansonsten FALSE.

- Milchstrasse.

⇒ Zweck: Erzeugt eine Menge von weißen Pixeln (Punktgröße 1.0) die zufällig im Viewing-Volumen angeordnet sind. Damit ergibt sich der Eindruck einer Milchstraße. Die Anzahl der Pixel und der Bereich, in dem diese angeordnet werden können, hängen von den übergebenen Parametern ab. Die Milchstraße ist als Displayliste realisiert.

⇒ Parameter: Anzahl der Sterne (Pixel) und maximale Ausdehnung des Raumes in dem diese angeordnet werden können.

⇒ Bemerkung: Die Performance steigt mit abnehmender Zahl der darzustellenden Pixel.

- Planet1.

⇒ Zweck: Erzeugt einen grünen Planeten mit dem Radius 0.6. Der Planet wird im Raum positioniert, wobei seine X- und Z-Koordinaten veränderlich sind. Dadurch ist eine Rotation um die Sonne möglich.

⇒ Parameter: Keine.

- Planet2.

⇒ Zweck: Erzeugt einen roten Planeten mit dem Radius 0.7. Zusätzlich erhält er zwei um ihn befindliche Ringe (Saturn). Um die Ringe für den Beobachter deutlicher erkennbar zu machen, wird die Position der Planetenachse verändert. Der entsprechende glRotatef()-Aufruf bezieht sich auf die aktuelle Matrix und hat somit auch Einfluß auf die Stellung der Ringe, welche erst nach diesem Aufruf erzeugt werden. Die Ringe sind gelb- bzw. ockerfarben. Bei ihnen muß darauf geachtet werden, daß die Rückseiten dargestellt werden, sonst würden sie (die Ringe) verschwinden, sobald der Beobachter hinten ihnen ist. Dies wird mit dem Aufruf glDisable(GL_CULL_FACE) gewährleistet. Der Planet wird im Raum positioniert, wobei seine X- und Z-Koordinaten veränderlich sind. Dadurch ist eine Rotation um die Sonne möglich.

⇒ Parameter: Keine.

- Planet3.

⇒ Zweck: Erzeugt einen cyanfarbenen Planeten mit Radius 1.8. Ansonsten siehe Beschreibung zu Planet1.

⇒ Parameter: Keine.

- Planet4.

⇒ Zweck: Erzeugt einen rot-grünfarbenen Planeten mit Radius 1.6. Es gilt das für Planet1 Geschriebene, jedoch wird bei diesem Planeten zusätzlich die Y-Koordinate während der Rotation um die Sonne verändert, da dieser Planet schief zur Ekliptik steht, d.h., seine Rotationsebene weicht von der der anderen Planeten ab.

⇒ Parameter: Keine.

- SetzePixelFormat.

⇒ Zweck: Diese Funktion stellt das Pixelformat so ein, daß die OpenGL-Funktionen in den angegebenen Gerätekontext zeichnen können und dies auch sichtbar wird. Dazu werden die beiden Funktionen ChoosePixelFormat() und SetPixelFormat() benutzt. Erstere sucht die beste Übereinstimmung die auf dem übergebenen Gerätekontext mit einer ebenfalls übergebenen Struktur vom Typ PIXELFORMATDESCRIPTOR möglich ist. Dazu muß diese Struktur zumindest den OpenGL-Support anfordern, weitere Einstellungen sind teilweise „egal“, können jedoch die Performance verbessern helfen. Ist die Funktion erfolgreich (ChoosePixelFormat()), dann gibt sie einen Wert ungleich 0 zurück, den dann die Funktion SetPixelFormat() benutzt, um dieses bestmöglich übereinstimmende Pixelformat auf dem entsprechenden Gerätekontext zu realisieren.

⇒ Parameter: Keine.

⇒ Rückgabe: Bei Erfolg TRUE ansonsten FALSE.

- Sonne.

⇒ Zweck: Erzeugt eine Sonne mit gelber Farbe und gelb emittierendem Licht und dem Radius 2.0. Sie wird im Zentrum der Szene (des Raumes) plaziert.

⇒ Parameter: Keine.

- Sonnenlicht.

⇒ Zweck: Erzeugt eine Lichtquelle, die weißes Licht abstrahlt. Sie wird im Zentrum (Koordinaten 0.0, 0.0, 0.0) des Raumes plaziert und gibt so der Sonne das Aussehen eines strahlenden Körpers.

⇒ Parameter: Keine.

- Sterne.

⇒ Zweck: Zweck und Beschreibung siehe Beschreibung zur Funktion Milchstrasse(). Hier ist nur die Größe der Objekte (Sterne) auf 3x3 Pixel gesetzt und die Farbe wird zufällig ermittelt.

⇒ Parameter: Keine.

Globals (globale Daten)

Es existiert nur die globale Variable „theApp“ vom Typ CAnimationenApp. Sie repräsentiert das Anwendungsobjekt.