# Multimedia-Projektworkspace

von jk-ware, Dipl.-Ing. Reinhard Jakob

Version 4.0 vom 15.05.2018

## 1 Vorwort

Mit dem Multimedia Projektworkspace wird das Produkt jk-ware Multimedia-Theater erstellt.

Ebenso können damit auch größere Projekte unter dem .Net Framework für Windows Forms realisiert werden. Hierzu werden die Anwendungen von einem umfassenden Basisprojekt abgeleitet.

Dieses Handbuch beschreibt die Projektmappe des Multimedia-Projektworkspace in einer Visual C# Entwicklungsumgebung für Windows Forms und managed DirectX und liefert Informationen zu den darin vorhandenen Projekten und deren Komponenten. Es enthält aber keine Abhandlung über die Programmiersprache C# oder über das .Net-Framework, soweit dies über die erforderlichen Erläuterungen zum Programmablauf hinausgeht.

Die Quelltexte sind nur in dem Handbuch des Endproduktes enthalten. Auch mit diesen kann ein umfassendes Verständnis des Programmcodes nur durch die kontextsensitive Hilfe innerhalb der Entwicklungsumgebung erlangt werden.

Hinweis 1 zu dieser Word-Dokumenterstellung !

- Die aufgeführten Inhaltspunkte sind im Text als Formatierungen Überschrift 1, Überschrift 2 und Überschrift 3 gekennzeichnet.
- Die Hauptkapitel mit der Formatierung Überschrift 1 beginnen durch Einfügen / Manueller Umbruch / Abschnittsumbruch auf ungeraden Seiten.
- Die Kapitelnummern werden dadurch nach einer Bearbeitung der Formatvorlage für die Überschrift 1 mit Anpassung der Gliederung über Auswahl von Nummerierung unter Format automatisch erzeugt. Sie können auch manuell als Gliederung der Überschriften unter Format / Nummerierung und Aufzählungszeichen zugefügt werden.
- Das Kapitelverzeichnis und das Inhaltsverzeichnis werden durch Einfügen / Referenz / Index und Verzeichnisse / Inhaltsverzeichnis erzeugt. Hierbei werden für das Kapitelverzeichnis eine Ebene und für das Inhaltsverzeichnis drei Ebenen ausgewählt.
- Über Einfügen / Referenz / Beschriftung werden neue Bezeichnungen „Kodierung" für die Quellcode-Titel mit einbezogenen Kapitelnummern und „Hinweis" mit fortlaufender Nummerierung für weiterführende Hinweise zugefügt.
- Zum Einfügen des Quellcodes wurden unter dem Format-Menü alle Tabstopps gelöscht und danach die Position der Standardtabstopps auf 0,5cm gesetzt.
- Die Standard-Formatvorlage wurde von Times New Roman 12Pt auf Verdana 10Pt verändert.
- Unterschiedliche Seitennummerierung wird durch Abschnittswechsel über Einfügen / manueller Umbruch / Abschnittsumbruch nächste Seite möglich. Jedem Abschnitt können danach separat formatierte Seitenzahlen zugewiesen werden.
- Eine Formatvorlage innerhalb der Kopfzeile wird über Ansicht / Kopfzeile (Kopfzeile bearbeiten) mit Einfügen / Schnellbausteine / Feld Kategorie „Verknüpfungen und Verweise" mit Feldname StyleRef und der Felderoptionen Formatvorlagen hinzugefügt. Soll nur die Nummer der Gliederungsebene (Kapitelnummer) angezeigt werden, ist zusätzlich \n mit anzugeben.
- Der Titeltext, dieser Hinweis, der Kommentar, das Inhaltsverzeichnis und weitere Hinweise innerhalb des Handbuches sind in einer einfachen Tabelle eingefügt. Über Format / Rahmen und Schattierungen kann danach die Hintergrundfarbe geändert werden.
- Alle Hinweise beziehen sich auf die Möglichkeiten und Menüauswahl von Word 2003 und 2010.
- Dem Dokument kann über Format / Design ein Hintergrundmuster zugefügt werden.
- Literaturquelle: Albrecht, Nicol: Wissenschaftliche Arbeiten Schreiben mit Word, Addison-Wesley 2002, ISBN 3-8273-1943-9, auch als kostengünstiges eBook.

# 2 Kapitelverzeichnis

# 3 Inhaltsverzeichnis

# 4 Kurzbeschreibung

Mit dem MDI-Projektworkspace für Visual C# stellt jk-ware seine Projektbasis für das .Net Framework zur Verfügung. Diese Projektbasis, bestehend aus fünf Projekten für die Entwicklerumgebungen Microsoft Visual C# ab 2012, bildet die Grundlage zur Entwicklung eigener Programme für das Laufzeitsystem des .Net Framework.

Der MDI-Projektworkspace besteht aus einem

- Basisprojekt das ein Basisprogramm erstellt
- Anwendungs-Projekt mit einem Beispielprogramm
- Multimedia Theater-Projekt zur Verwaltung der Formen
- Projekt zur Erstellung eines kryptographischen Aktivierungsschlüssels
- Projekt zur Einbindung eines erweiterten Lizenzschutzes

## 4.1 Das Basisprojekt

Das Basisprojekt stellt eine Grundlage für die Erweiterung von Programmen auf die .Net-Entwicklungsplattform zur Verfügung. Die zu realisierenden Anwendungen werden alle von diesem Basisprojekt abgeleitet und erben dadurch grundlegende Eigenschaften.

Das aus dem Basisprojekt durch den Compiler erzeugte Basisprogramm realisiert ein Ausgabeformular (Form) mit einem Auswahl- und einem Kontextmenü. Die Grafik des Basisprogramms gibt einfache geometrische Figuren in eine Basisbitmap und zur Grafikfläche der Form aus.

Über das Auswahlmenü kann über das Basis-Menüelement

- eine modale Dialogform mit drei Seiten aufgerufen werden:
  - „Basis" enthält grundlegende Einstellungen für das Basisprogramm wie Hauptmenü- und Statusleiste ein- oder ausblenden, die Sichtbarkeit der Form und transparenter Hintergrund.
  - „Grafik" mit grundlegenden Einstellungen für die Grafikausgabe wie SmoothingMode und PixelOffsetMode.
  - „Sprache" beinhaltet die Sprachauswahl für die Programmbedienung und das Hilfesystem.
- eine Grafik, Audio- oder Videodatei geladen und hiervon die Originalgröße ausgewählt werden
- die Grafikdatei in verschiedenen Formaten abgespeichert und ausgedruckt sowie
- die Statusleiste ein- und ausgeblendet werden.

Über das „Hilfe"-Menü kann die HTML-Hilfe für das Programm und die Menüauswahl aufgerufen und die Sprache für die Programmbedienung und das Hilfesystem ausgewählt werden. Zur Sprachauswahl stehen Deutsch, Englisch, Französisch, Italienisch, Spanisch und Portugiesisch.

## 4.2 Verwendung des Basisprojekts

Neue Projekte werden von der BASISPROJEKT-Klasse abgeleitet.

Der Konstruktor kann dreifach aufgerufen werden: Ist der erste Parameter ein string, wird ein Kontrollablauf zur Abfrage eines Passwortes oder einer Lizenzdatei durchgeführt. Für die Abfrage einer Lizenzdatei ist der zweite Parameter true zu setzen.

Durch eine weitere Ableitung mit dem ersten Parameter true, wird der Licence Protector von Mirage-Systems eingebunden. Hierzu muss für das Basisprojekt unter Symbole für bedingte Kompilierung auch die Stringkonstante MirageLP zugefügt werden.

Zur Realisierung der nachfolgenden Punkte werden die Quellcode-Dateien des Basisprojekts (Basisprojekt.cs, Basis_Globals.cs, Basis_Procs.cs, Basis_Dialog.cs, Basis_Dialog_Texte.cs, ManagedDirectX.cs und WAPI_CodePack.cs) als vorhandene Elemente mit dem abgeleiteten Projekt unter dem neu erstellten Ordner Basisprojekt verknüpft. Hierzu müssen die Dateien über den rechten Pfeil der Taste „Hinzufügen" mit „als Verknüpfung hinzufügen" ausgewählt werden.

Wenn keine Multimediadatei angezeigt wird und keine Programmpause ist, wird in BASISPROJEKT.Basis die Methode Programm ständig aufgerufen. Programm ist virtuell definiert und kann somit durch eine gleichnamige Methode der Anwendung überschrieben werden.

Zur Abarbeitung anstehender Ereignisse muss das Programm ständig verlassen werden. Dies ist auch bei den Multithread-Versionen zur Synchronisation der Ereignisse mit dem Programmthread beim Vergrößern und Neuzeichnen der Form erforderlich.

Das Basisprogramm erstellt ein Hauptmenü mit den Punkten „Programm", „Basis" und „Hilfe" und ein Kontextmenü, das auf Betätigung der rechten Maustaste angezeigt wird.

Die Anwendung erweitert das Basismenü durch Überschreiben der virtuellen Methode ErzeugeFormMenues um neue Punkte oder verändert bestehende Punkte beider Menüs. Hierzu wird vorab die überschriebene Methode des Basisprojekts aufgerufen.

Danach können die einzelnen Menüelemente über die öffentlichen Arraytexte direkt verändert und erweitert werden. Hierzu müssen auch die Dimensionen der Textarrays angepasst werden.

Der Titeltext wird in der gleichnamigen virtuellen Methode ermittelt und gesetzt. Der Aufruf erfolgt in ErzeugeFormMenues. Hierdurch wird er auch nach einer Sprachauswahl korrekt verändert. Die Anwendung überschreibt diese virtuelle Methode, wodurch der korrekte Text für alle Versionen der Anwendungen polymorph ermittelt wird. Bei Dialogformen wird der ermittelte Titeltext des Besitzer-Programms an den Dialogformtext angehängt.

Zum direkten Zeichnen auf dem Display wird ein GDI+ Graphics-Objekt in der Größe des Panels zur allgemeinen Verwendung zur Verfügung gestellt. In der gleichen Größe wird für die Hintergrundgrafik eine Basisbitmap mit einem Graphics-Objekt erzeugt.

Zur Aktualisierung der Grafikfläche der Form wird in PanelOnPaint die vom Basisprogramm erzeugte Basisbitmap verwendet. Hierzu muss die Anwendung in der Basisbitmap immer die aktuelle Programmgrafik zur Verfügung stellen. Diese kann dann auch zum „Drucken..." und „Speichern..." verwendet werden. Die Anwendung zeichnet eine Raumgrafik mit Linien und Farben in die Basisbitmap.

Zum Projekt der Anwendung gehört die ANWENDUNG1-Klasse und die im Konstruktor erzeugten Basisklassen ANW1_GLOBALS für öffentliche Projektdaten und EVENT_PROCS. EVENT_PROCS besteht aus Methoden, die innerhalb der Botschaftsmethoden aufgerufen werden. In ihnen bearbeitet die Anwendung das Ereignis und zeichnet die Raumgrafik.

Die Dialogform der Anwendung mit drei Seiten (Tab-Pages), wird in ANW1_DIALOG erstellt. Die hierzu erforderlichen Texte werden in ANW1_DIALOG_TEXTE erzeugt.

**Hinweis 2 zur Kompilierung der Einzelprogramme!**

Die Kompilierung der **Einzelprogramme** muss in der Entwicklungsumgebung über Eigenschaften für die Anwendung und das Basisprojekt unter Erstellen die Stringkonstante für bedingte Kompilierung **MDI_PRJ** unbedingt entfernt werden. Diese ist auch bei der Kompilierung des MDI-Projekts nicht unbedingt erforderlich. Hierdurch sind aber die Anweisungen für das MDI-Projekt immer grau schattiert.

**Hinweis 3 zum Ablauf der Einzelprogramme!**

Für den Ablauf der **Einzelprogramme in einem Thread** wird anstelle des Arbeitsthreads ein Timer gestartet. Dieser aktiviert den Programmablauf innerhalb der Timer-Methode. Hierzu muss für die Anwendung und das Basisprojekt unter Kompilierungseigenschaften auch der bedingte Kompilierungsstring **MULTITHREAD** entfernt werden. In der durch den Timer aktivierten Methode Programmschleife(...) werden dann zusätzlich ständig die Programmereignisse abgearbeitet. Die Abfrageanzahl der Programmereignisse in jeder Sekunde kann über die Variable EreignisAbfrageMax begrenzt werden (Standardwert 50). Hierdurch steht dem Programm die Computer-Leistung auch kontrolliert zur Verfügung.

## 4.3 Der Ablauf

Zum Programmstart wird im BASISPROJEKT-Konstruktor eine Instanz von BASIS_GLOBALS mit den öffentlichen Datenfeldern des Basisprogramms erzeugt.

Bevor die Form angezeigt wird, werden danach auf OnLoad folgende Abläufe durchgeführt:

- Zur Protokollierung der Programmausnahmen wird eine globale StreamWriter-Instanz SWriter für eine Textdatei erzeugt. Der Stream wird erst auf OnClosed wieder geschlossen und steht somit im gesamten Programmablauf für die Ausnahmeprotokollierung zur Verfügung.
- die letzten Einstellungen des Programms werden in Programm_Init zurück geladen. Für das MDI-Projekt wird zum Dateinamen die ermittelte Programmnummer zugefügt.
- eine Panel-Kontrolle mit Ziehleisten, eine Statusbar und das Haupt- und Kontextmenü für die Form werden erzeugt.
- ein Arbeitsthread mit der Startmethode MT_Programmschleife wird gestartet. In diesem Arbeitsthread wird zur Realisierung des Programmablaufs die Methode Basis ständig aufgerufen. Basis wiederum aktiviert die von der Anwendung zu überschreibende Methode Programm. Der Arbeitsthread bleibt so lange aktiv, bis zum Programmschluss in OnClosed die Schleifenbedingung zu false gesetzt und MT_Programmschleife verlassen wird.
- Für die Einzelversionen der Programme mit nur einem Bedienerthread wird ein Zeitgeber zur Aktivierung der Programmschleife aktiviert.

Für die Multithreadversion der Einzelprogramme wird in der durch den Timer aktivierten Programmschleife nur ein Kontrollablauf durchgeführt. Danach wird diese Programmschleife sofort wieder verlassen. Dieser Ablauf ist erforderlich geworden, weil die zum Bedienerthread gehörenden Dialogformen nicht innerhalb der Startmethode des Arbeitsthreads MT_Programmschleife erzeugt werden dürfen (weitere Erläuterungen hierzu im folgenden Abschnitt).

Wenn am Anfang der Programmschleife in einer binären Kontroll- oder Lizenzdatei kein gültiger String vorhanden ist, werden innerhalb des Kontrollablaufes zur Passwortabfrage oder zur Anforderung der Lizenzdatei mehrere Dialogformen angezeigt.

Der Ablauf zur Anforderung der Lizenzdatei im Einzelnen:

- Nach dem Dialog zur Sprachauswahl wird der Lizenzdialog mit dem Lizenzvertragstext (EULA) in der ausgewählten Sprache angezeigt. Wenn dieser nicht akzeptiert wird, wird der weitere Ablauf abgebrochen und es läuft nur die Testversion des Programms.
- Wird der Lizenzvertrag akzeptiert, kann der Kunde innerhalb der nächsten Dialogform über Radiobutton auswählen, ob die Registrierung für alle Benutzer des Computers oder nur für den gerade angemeldeten Benutzer gültig sein soll. Der ausgewählte Registrierungstext wird in die Zwischenablage kopiert und muss dann per E-Mail an den Verkäufer gesendet werden. Der Registrierungstext besteht aus dem Produktnamen mit Versionsnummer und

dem Dateipfad der Anwendung für alle Benutzer oder für den gerade aktuellen Benutzer mit dem Erzeugungsdatum des Ordners.
- Nach Zahlungseingang wird mit dem kleinen Zusatzprogramm aus dem CryptoKeyGenerator-Projekt aus dem Registrierungsschlüssel zusätzlich mit einem internen Projektpasswort der Aktivierungsschlüssel erzeugt. Dieser wird in einer Lizenzdatei an den Kunden zur Freischaltung der Anwendung übermittelt.

Nachdem die Form sichtbar ist und nach jeder Änderung der Formgröße wird auf PanelOnResize ein Grafik-Objekt für die Form und eine Basisbitmap mit dem zugehörigen Grafik-Objekt erzeugt.

Zur Anzeige eines Bildes oder Ausgabe einer Grafik die größer als die Grafikfläche der Form ist, muss die Anwendung die Basisbitmap mit dem zugehörigen Grafik-Objekt in der erforderlichen Größe selbst erzeugen. Hierzu muss die öffentliche Instanzvariable Grafik_Anzeige oder Multimedia_Anzeige true gesetzt werden. In PanelOnResize wird dann keine Basisbitmap erzeugt.

Zur Ablaufkontrolle zeichnet das Basisprogramm einfache geometrische Figuren in die Basisbitmap und zum Display. Die Anwendung stellt verschiedene Raumgrafiken zur Auswahl und zeichnet auf dem Raumboden geometrische Figuren.

Durch Auswahl des Menüpunktes „Dialog" der Anwendung wird eine modeless Dialogform mit zwei Seiten angezeigt: auf der Basis-Seite wird die Aufruffrequenz der Anwendung durch einen auf OnLoad erzeugten Sekunden-Timer ausgegeben und ein Button zur Information über die Anwendung eingefügt. Die Grafik-Seite ermöglicht die Auswahl der Raumgrafik.

## 4.4 Der Ablauf mit mehreren Threads

Eine erzeugte Programm-Instanz startet im Bedienerthread. Aufgrund der Abarbeitung der Botschaften im Bedienerthread wird dieser im weiteren Ablauf auch Botschaftsthread genannt oder auch als User-Interface-Thread (UI-Thread) bezeichnet. Bei einem Ablauf mit mehreren Threads erzeugt das Programm zusätzlich zu seinem Bedienerthread explizit einen oder mehrere Arbeitsthreads.

Gleichzeitig zu den Botschaften für die Form verrichtet der Arbeitsthread weitere Aufgaben. Aus diesem Grund darf der Arbeitsthread nicht auf Objekte zugreifen, wenn diese gleichzeitig im Botschaftsthread verwendet oder neu erstellt werden. Um dieses zu Vermeiden, stellt der **Bedienerthread** in ThreadSynchroStart eine Synchronisationsanforderung an den Arbeitsthread und wartet solange bis dieser diese bestätigt. Nach dem Objektzugriff im Bedienerthread wird in dem darauf erforderlichen Aufruf von ThreadSynchroEnde die Synchronisationsanforderung wieder zurückgesetzt.

Eine Synchronisierung des Bedienerthreads mit dem Arbeitsthread ist erforderlich:

- In PanelOnResize des Basisprojekts vor dem Erzeugen eines neuen Grafikobjekts und der Basisbitmap.
- Auf PanelOnPaint des Basisprojekts vor der Verwendung der Basisbitmap.
- In OnClosed für den Basis-Dialog vor der Zuweisung der Daten.
- Auf PanelOnResize der Anwendung vor dem Neuzeichnen des Hintergrundes in die Basisbitmap in Groessen.

Komponenten und somit auch Dialogformen müssen im Bedienerthread erzeugt werden. Nur hierdurch können die im Bedienerthread ablaufenden Methoden auf die darin vorhandenen Steuerelemente zugreifen!

Vor der Erzeugung von Komponenten sollte deshalb abgefragt werden, ob die Erzeugermethode im Arbeitsthread ausgeführt wird und somit auf den Bedienerthread umgeleitet werden muss. Zur Umleitung wird die Erzeugermethode durch eine der Invoke-Methoden synchron oder asynchron als Delegat ausgeführt.

Delegaten werden in allen Projekten am Anfang der Methoden zur Erzeugung der Menüs erzeugt und ausgeführt. Des Weiteren im Basisprojekt in Animated_Bitmap zur Animation von geladenen Bitmaps.

Die Bedingte Kompilierungskonstante **MULTITHREAD** ermöglicht den Ablauf der Programme mit mehreren Threads. Die Kompilierung mit **MULTITHREAD** ist für das MDI-Projekt unbedingt erforderlich. Die Einzelversionen können mit und ohne **MULTITHREAD** kompiliert werden.

Weitere Anweisungen mit Bedingter Kompilierung innerhalb von #if MULTITHREAD ... #endif außerhalb des MDI-Projekts:

- Alle using-Anweisungen, die den System.Threading-namespace einbinden.
- Die Referenz auf den Arbeitsthread und Basisprojekt-Instanzvariable zur Synchronisierung der Threads.
- Die Startmethode für den erzeugten Arbeitsthread MT_Programmschleife mit den Synchronisationsmethoden ThreadSynchroStart und ThreadSynchroEnde.
- In OnLoad wird im Basisprojekt der Arbeitsthread mit MT_Programmschleife als Startmethode erzeugt.
- Zum Verlassen von Programmschleife für die Einzelversionen nach einem erforderlichen Kontrollablauf.
- Nachdem in OnClosed die Schleifenvariable false gesetzt wurde, wird vor der Freigabe von Programmobjekten das Ende des Arbeitsthreads abgewartet.

## 4.5 Die Verwaltung der MDI-Formen

Die Verwaltung der MDI-Formen wird in der Projektmappe „MDI_Projekt" realisiert.

Zu den Eigenschaften von MDI_Projekt **muss** zur bedingten Kompilierung neben **MULTITHREAD** auch noch die Stringkonstante **MDI_PRJ** zugefügt werden.

Das MDI-Projekt stellt ein Elternformular zur Anordnung und Verwaltung der Programme als MDI-Formen zur Verfügung.

Im Konstruktor wird eine Basisklasse MDIFORM_GLOBALS mit öffentlichen Instanzvariablen und Basis- und Menütexten erzeugt und die für die Designerunterstützung erforderliche Methode InitializeComponent aufgerufen.

Auf OnLoad werden in Programm_Init die letzten Einstellungen des MDI-Verwaltungsprogramms zurück geladen sowie die Statusleiste und das Menü erzeugt. Dieses Basismenü mit den Punkten Programm, Formen und Hilfe wird mit den Menüpunkten der aktivierten MDI-Form zum Hauptmenü erweitert.

Eine binäre Kontroll- oder Lizenzdatei kann in Kontrollmethode abgefragt werden. Wenn eine gültige Lizenzdatei vorhanden ist, wird die öffentliche Variable TESTVERSION false.

Kontrollmethode wird durch einen in OnLoad erzeugten Timer aufgerufen und läuft somit im Bedienerthread. Dies ist zur korrekten Anzeige der Dialogformen während des Kontrollablaufes über einer sichtbaren Form unbedingt erforderlich.

Unter dem Programmmenüpunkt werden die zu verwaltenden MDI-Programme gestartet und beendet. Zur Verfügung stehen die vom Basisprojekt abgeleitete Anwendung und das Basisprogramm selbst.

Unter dem Formen-Menüpunkt werden die MDI-Formen aufgelistet und angeordnet. Hierunter können diese auch einzeln oder alle gleichzeitig geschlossen werden.

Bei jeder Erzeugung einer neuen Anwendung wird die Programmnummer auf ProgrammMenueOnClick ermittelt, bevor für die MDI-Form des Programms durch Show auf OnLoad des Basisprogramms der Titeltext zusammengestellt wird.

Neben der Sprachauswahl und dem Datenfeld TESTVERSION wird die Programmnummer den öffentlichen Daten des gestarteten Programms übergeben.

Wenn die aktive MDI-Kindform bei einer geöffneten Dialogform der Anwendung gewechselt wird, werden in OnMdiChildActivate die Referenzen auf die öffentlichen Instanzvariablen der Anwendung, die dem Konstruktor der Dialogform übergeben wurden, gegen die öffentlichen Referenzen der aktivierten Anwendung ausgetauscht. Danach wird die Dialogform durch Aufruf von LoadHandler neu aufgebaut.

Bedingte Anweisungen innerhalb von #if MDI_PRJ ... #endif:

- In Programm_Init des Basisprojekts wird zum übergebenen Filename die Programmnummer zugefügt. Die Datenfelder User_Language und Help_Language werden nicht für die Init_Datei berücksichtigt, weil die Sprachauswahl vom MDI-Programm zugewiesen wird.
- In Titeltext der Anwendung und des Basisprogramms zur Verknüpfung der Programmnummer mit dem Text.
- In LoadHandler für den Text der Dialogform der Anwendung und in OnLoad für die Basisdialogform.
- In ProgrammMenueOnClick und in RaumgrafikMenueOnClick für die Anwendung zum Suchen nach einer Anwendung mit geöffneter Dialogform.
- Nach Menüauswahl „Beenden !" in ProgrammMenueOnClick für die Anwendung, wird nicht Application.Exit ausgeführt, da hierdurch das Formen-Verwaltungsprogramm beendet wird.
- In ErzeugeFormMenues der Anwendung zur korrekten Anzeige des Raumgrafik-Menüpunktes nach einem Sprachwechsel.
- In OnLoad des Basisprojekts zur Erzeugung der öffentlichen StreamWriter-Instanz.

## 4.6 Abspielen und Steuern von Video- und Audiodateien

In BASISPROJEKT.Image_Audio_Video_Laden werden Videodateien in der Panelform visuell und Audiodateien akustisch wiedergegeben. Hierzu wird das AudioVideoPlayback API von Managed DirectX eingesetzt.

Die immer eingeblendete horizontale Ziehleiste wird zur Positionssteuerung der Video- und Audiodatei in BASIS_PROCS.HorzRollbalken verwendet. Die Steuerung mit der MediaPlayPause-Taste und der Pause-Taste erfolgt in BASISPROJEKT.OnKeyDown. Zur Feinsteuerung der Videoposition wird die linke und rechte Pfeiltaste in Verbindung mit der Steuertaste verwendet.

Die Position der Video- und Audiodatei wird danach auf BASISPROJEKT.PanelOnPaint durch den **set-Wert** der CurrentPosition-Property gesetzt (durch die SeekCurrentPosition-Methode wird das Video immer nur neu gestartet). Hierin wird auch ein Bild oder Video an die Panelgröße angepasst, wenn es nicht in Originalgröße angezeigt werden soll.

Zur Aktualisierung der horizontalen Ziehleiste und der Paneltexte der Statusleiste wird ein Basiszeitgeber in BASISPROJEKT.OnLoad eingerichtet und in Image_Audio_Video_Laden gestartet. In der Zeitgebermethode Basis_Zeitgeber wird die horizontale Ziehleiste mit der Position des Videos oder Audios aktualisiert. Des Weiteren werden die erforderlichen Panels verwaltet und ihnen ein Text und Tooltiptext zugewiesen.

Nach dem Programmstart wird die letzte Multimediadatei wieder geladen, wenn das Programm danach nicht mehr aktiviert wurde. Hierzu werden die Datenelemente Multimedia_Anzeige und DialogFileName in die Initialisierungsdatei des Programms abgespeichert.

Für das Abspielen von Audio- und Videodateien wird die Bedingte Kompilierungskonstante „DAUDIOVIDEO" eingeführt. Nur bei einer fehlenden Angabe dürfen alle vom Basisprojekt abgeleiteten Programme alle Plattformen der Projektmappe verwenden.

## 4.7 Der Aktivierungsschlüssel

Im CryptoKeyGenerator-Projekt wird aus dem Registrierungsschlüssel und einem Projektpasswort eine binäre Lizenzdatei "jk-ware_Lizenzdatei.bin" mit dem Aktivierungsschlüssel erstellt.

Hierzu stellt das Projekt eine Form als FixedToolWindow zur Verfügung. Innerhalb der Form wird in zwei Textboxen der von den Programmen erzeugte Registrierungsschlüssel und das Projektpasswort eingegeben oder aus der Zwischenablage eingeholt. Daraus wird über die Taste „Erzeugen" eine Lizenzdatei mit dem Aktivierungsschlüssel generiert.

Die Programme überprüfen in der Kontrollablauf-Methode der LIZENZSCHUTZ-Klasse den Aktivierungsschlüssel in der Lizenzdatei. Ist die Lizenzdatei nicht vorhanden oder der Aktivierungsschlüssel nicht korrekt, wird als ein Angebot zum Erwerb einer Lizenzdatei der Kontrollablauf durchgeführt.

## 4.8 Allgemeine Hinweise zur Kodierung

- Für die Übersetzungen des MDI_Projektworkspace wurde Microsoft Visual Studio Community 2017 Version 15.4.1 mit Microsoft .NET Framework Version 4.7.02046 und Visual C# 2017 verwendet.
- Allen Projekten wird als ausreichendes Zielframework .Net Framework 4 Client Profile zugewiesen.
- Zur Verwendung der Managed DirectX-Bibliothek muss in den Projekt-Eigenschaften unter Erstellen die x86-Zielplattform ausgewählt werden (s.a. Die Managed DirectX-Bibliothek).
- Zur Erstellung von CLS-kompatiblen Code (CLS = Common Language Specification) wird dem Basisprojekt die Anweisung [assembly: CLSCompliantAttribute(true)] zugefügt. Allen Klassen muss des Weiteren die Anweisung [CLSCompliantAttribute(true)] vorangestellt werden. Zur korrekten Ausführung durch den Compiler müssen die Klassen als public deklariert sein.
- In InitializeComponent werden durch Neuzuweisungen der Formgröße implizit OnResize- und MdiChildActivate-Ereignisse ausgelöst. Um einen Ausnahme-Fehler, z.B. wegen noch nicht erzeugter Objekte zu vermeiden, wird der Aufruf unter OnLoad kontrolliert eingefügt.
- Die Klassen mit öffentlichen Datenfeldern für die Anwendung und für das Basisprojekt erben von Klassen mit multilingualen Texten. Hierzu gehören allgemeine Projekttexte, Texte für Fehlermeldungen in Messageboxen und Menütexte. Weil diese als static string-Arrays deklariert sind, werden innerhalb der MDI-Formenverwaltung nicht für jede gestartete Anwendung öffentliche Texte erzeugt. Die Texte sind dadurch nur einmal im Speicher vorhanden, werden aber für jede Instanz der öffentlichen Daten neu initialisiert.
- Die Grafikfläche der Form wird als Panel organisiert. Hierdurch werden die Ziehleisten korrekt mit der Statusleiste in die Form integriert und eine geteilte Form ermöglicht.
- Bei der Erweiterung des Hauptmenüs darf der Basismenüpunkt nicht verändert werden. Bei einer Veränderung der Position des „Pause !"-Menüpunktes unterhalb von „Programm", ist innerhalb der Eigenschaft für diese Instanzvariable die Checkmarke neu festzulegen. Dies gilt ebenso für die Checkmarke der Statusleiste, die in der Eigenschaft für die Instanzvariable StBarAnzeige verändert wird.
- Für einen einfachen Zugriff auf die Ressourcen des Basisprogramms und der Anwendung wurde der Standardnamespace unter den Projekt-Eigenschaften entfernt.

## 4.9 Literaturquellen


1. Moses, Nowak: C# Programmieren unter .Net, Franzis' 2002, ISBN 3-7723-7224-4: empfehlenswert zur Einführung in die Programmiersprache C#.

2. Petzold: Programming Windows with C#, Microsoft 2002, ISBN 0-7356-1370-2: empfehlenswert zur Einführung in die Form-Programmierung unter .Net.

3. Schwichtenberg, Eller: Programmierung mit der .Net-Klassenbibliothek, Addison-Wesley 2002, ISBN 3-8273-1905-6: zusätzlich zur Beschreibung der .Net-Klassenbibliothek von Microsoft wird als eBook ein kostengünstiger Überblick geliefert.

4. Maslo, Freiberger: .Net-Framework Developer's Guide, Markt&Technik 2002, ISBN 3-8272-6142-2: zusätzlich zur Beschreibung der .Net-Klassenbibliothek von Microsoft wird als eBook ein kostengünstiger Überblick geliefert.

5. Sells: Windows Forms Programming in C#, Addison-Wesley 2004, ISBN 0-321-11620-8: beschreibt tiefer gehend die Probleme der .Net-Programmierung für MDI- und Multithread-Umgebungen.

6. Bayer: Das C#-Codebook, Addison-Wesley 2006, ISBN 3-8273-2326-6, mit nützlichen C#-Codes unter .Net auch als kostengünstiges eBook.

7. Jones: Programmierrezepte für Visual C# .Net, Microsoft Press Deutschland, ISBN 3-86063-091-1, mit nützlichen Programmierrezepten zu Visual C# .Net.

8. Schildt: C# IT-Tutorial, mitp-Verlag, ISBN 3-8266-0834-8, zur Auffrischung der C#-Grundlagen.

9. Kühnel: Das umfassende Handbuch zu Visual C# 2008 und Visual C# 2010, Galileo Computing als <openbook>

10. Miller: Managed DirectX9 Kick Start, SAMS, ISBN 0-672-32596-9

# 5 Die Projektentwicklung

## 5.1 Die Projektmappe

Die Projektmappe für den Multimedia Projektworkspace enthält fünf Projekte unter Microsoft Visual Studio Community 2017 Version 15.54. Die hieraus erstellte Microsoft Visual Studio Solution-Datei Multimedia_Projektworkspace.sln wird von Visual Studio ab 2012 und auch von nachfolgenden Entwicklungsumgebungen akzeptiert.

Zur Kompilierung der Projekte wurde laut Anzeige des Info-Dialogs unter dem Hilfe-Menüpunkt von Microsoft mit Microsoft .NET Framework Version 4.7 und Visual C# 2017 verwendet. Unter Anwendung der Projekteigenschaften werden für alle fünf Projekte als Zielframework .NET Framework 4 Cient Profile eingesetzt.

Die fünf Projekte sind in alphabetischer Reihenfolge die Anwendung1, das Basisprojekt, der CryptoKeyGenerator, Licenceprotectorsample und Multimedia Theater.

Ab der Version 4.0 wird das Abspielen von Audio- und Videodateien mit der Managed DirectX AudioVideoPlayBack-API realisiert. Hierfür muss für das Projekt als Zielplattform „x86" ausgewählt werden. Das Programm ist somit nicht mehr durch „Any CPU" plattformneutral (s.a. Die Bibliothek in das Projekt einbinden). Deshalb wird über den Konfigurationsmanager allen Projekten die x86-Plattform zugewiesen. Hierdurch werden für die binären- und objekt-Ausgabedateien unter „bin\x86" und „obj\x86" neue Ordner erstellt.

## 5.2 Programmierung in C#

Die Programme sollen die Anforderungen für alle vorhandenen und zukünftigen .Net-Plattformen erfüllen und einen Ablauf unter der Common Language Runtime ermöglichen. Hierzu ist allen Klassendeklarationen das [CLSCompliantAttribute(true)] vorangestellt.

Der Quellcode wird mit #region und #endregion in Regionen unterteilt. Hierdurch wird ein schneller Zugriff auf die einzelnen Programmabschnitte möglich.

Die Beschreibungen für alle Klassen, Methoden und öffentlichen Variablen sind im Quellcode über den Deklarationen als XML-Kommentar aufgeführt. Zur weitergehenden Anzeige im Objektkatalog und für die kontextsensitive Hilfe wurde der XML-Kommentar für alle Methoden mit <para>-Tags innerhalb der <summary>- und <remarks>-Tags versehen.

Neben der Erhöhung der globalen Variablen catch_Ausnahmen, werden die innerhalb des Programmablaufes auftretenden Ausnahmen mit Datum und Uhrzeit in Ausnahmeprotokollierung der BASISPROJEKT-Klasse in eine Datei geschrieben. Hierzu wird im Basisprogramm auf OnLoad ein Ausgabestream für die Datei „Titeltext" + "_Exceptions.txt" erzeugt, der auf OnClosed wieder geschlossen wird (für das MDI-Projekt heißt die Ausnahmedatei für alle als Formen gestarteten Programme immer „MDIFormen_Exceptions.txt").

Zum globalen Abfangen und Protokollieren von unbehandelten Ausnahmen werden in BPRJMAINCLASS.Main zwei Ausnahmehandler erzeugt, nach deren Aktivierung die Anwendung fortgeführt wird.

## 5.3 Die Ressourcen

Alle Ressourcen werden zur Absicherung in ein gleichnamiges Verzeichnis innerhalb des MDI_Projektworkspace abgelegt.

Zur Einbindung in das Projekt müssen die *.resx-Dateien geöffnet und die erforderlichen Ressourcen darin eingefügt werden. Hierdurch werden die Dateien nochmals in ein von Visual C# erzeugtes Projekt-Verzeichnis „Resources" abgelegt.

Diese Ressourcen werden in die Assemblies der Projekte eingebunden. Es handelt sich vorerst nur um Symbole (Iconen) und Bitmaps (Images).

Der Zugriff auf die Ressourcen erfolgt in dem Programm über die Anweisungen

```
System.ComponentModel.ComponentResourceManager resources = new
System.ComponentModel.ComponentResourceManager(typeof(jkBPrj.BASISPROJEKT));
Icon = resources.GetObject("Basis1") as Icon;
Image = resources.GetObject("logoPictureBox.Image") as Image;
```

Zur Ermittlung der Type-Klasse der einzuholenden Ressource kann folgende Anweisungsfolge verwendet werden:

```
System.Reflection.Assembly assem = System.Reflection.Assembly.GetExecutingAssembly();
foreach (string resourceName in assem.GetManifestResourceNames() )
   MessageBox.Show (resourceName, Text);
```

Hierdurch werden die Namen aller in den Assemblies vorhandenen Ressourcen in einer Messagebox ausgegeben.

Weitere Zugriffsmöglichkeiten bieten folgende Anweisungen

```
System.Reflection.Assembly assem = System.Reflection.Assembly.GetExecutingAssembly();
System.Resources.ResourceManager resman = new
      System.Resources.ResourceManager("jkBPrj.BASISPROJEKT", assem);
string ResText = (string)resman.GetObject("TestString1");
MessageBox.Show (ResText, Text);
Icon BasisIcon = resman.GetObject("Basis1") as Icon;
```

Oder mit einer Stream-Anweisung

```
System.IO.Stream stream = assem.GetManifestResourceStream("Anw1.ico");
if (stream != null)
{
      BPrj.Icon = new Icon(stream);
      stream.Close();
}
```

Für einen erfolgreichen Zugriff als Stream, sind die Ressourcen-Dateien über die Buildvorgang-Eigenschaft dem Projekt explizit als „eingebettete Ressource" anzugeben.

**Achtung:** Wenn die „eingebetteten Ressourcen" **zusätzlich** in einer .resx-Datei aufgeführt sind, werden diese Dateien zweimal in die .exe-Datei der Anwendung eingebunden, wodurch sich die Dateigröße u.U. merklich erhöht.

**Hinweis 4 zum Icon der Anwendung!**

Das Anwendungsicon nimmt eine Sonderstellung unter den Ressourcen ein. Es muss in den Projekteigenschaften der Anwendung unter Symbol und Manifest explizit angegeben werden. Hierdurch wird vom Compiler im Projektordner nochmals eine Kopie des Icons abgelegt, die nicht entfernt werden darf.

## 5.4 Die Konfigurationsdatei der Anwendung

Unter Visual Studio muss in der Projektdatei app.config zur Einbindung von Managed DirectX folgender Eintrag vorhanden sein:
<startup useLegacyV2RuntimeActivationPolicy="true">
<supportedRuntime version="v4.0"/></startup>

Für die automatische Größenänderung von Steuerelementen für HighDpi-Anzeigen muss der Konfigurationsdatei app.config
<appSettings><add key="EnableWindowsFormsHighDpiAutoResizing" value="true" /></appSettings>

zugefügt werden.

## 5.5 Der Entwurf der Programm-Formen

Die Programm-Formen werden manuell ohne Designer-Unterstützung entworfen.

Dies betrifft insbesondere den Entwurf der multilingualen Menüs und deren Erweiterungen durch die abgeleiteten Anwendungen.

Weitere Probleme in Visual Studio (Express):

- die Designer-Ansicht für die Programm-Formen steht nicht immer zur Verfügung.
- Im DEBUG-Ablauf entsteht beim Schließen der Designer-Ansicht für die Anwendung1 durch die Grafikausgabe eine sich wiederholende catch-Ausnahme. Diese wird verhindert, wenn in BASISPROJEKT.Programm keine Messagebox ausgegeben wird.
- der Designer übernimmt die Ressourcenverwaltung, wodurch selbst verwaltete Ressourcen (Icon) entfernt werden.

### 5.5.1 InitializeComponent

Als exemplarisches Beispiel wird hier InitializeComponent des Basisprojekts aufgeführt.

Weil die Form des Basisprogramms mit den Menüs manuell codiert wird, darf der Inhalt der Methode manuell mit dem Code-Editor verändert werden. Auf die Designerunterstützung wird wegen der aufgeführten Probleme verzichtet.

Der Aufruf erfolgt wegen eines implizit ausgeführten OnResize-Ereignis nicht im Konstruktor.

**Kodierung 5-1: BASISPROJEKT.InitializeComponent**

## 5.6 Der Entwurf der Dialogformen

Alle Dialogformen werden mit Designer-Unterstützung entworfen.

Die Kodierung wird automatisch durch den Designer in InitializeComponent erstellt und deshalb in diesem Handbuch nicht aufgeführt.

# 6 Das Basisprojekt

Das Basisprojekt stellt eine Form mit vielfältigen Eigenschaften zur Vererbung bereit. Diese Eigenschaften sind

- ein Hauptmenü
- ein Kontextmenü
- eine Statusleiste
- ein Panel mit automatischen Ziehleisten
- Drucken der Programmgrafik
- Speichern der Programmgrafik
- Laden einer Grafik-, Audio- und Videodatei
- Lizenzschutz

## 6.1 Die Klassen

Das Basisprojekt besteht aus den Klassen BPRJMAINCLASS, BASISPROJEKT, BASIS_GLOBALS, BASIS_TEXTE, BASIS_MENUE_TEXTE, den Klassen für die Basisdialogform BASIS_DIALOG, BASIS_DIALOG_TEXTE, Dialogformklassen zur Informationsausgabe ABOUTBOX, UEBERDIALOG und INFO_DIALOG und den LIZENZSCHUTZ-Klassen mit den Dialogformklassen SPRACHE_DIALOG, LIZENZ_DIALOG, PASSWORT_DIALOG, REGISTRIERUNGS_DIALOG und AKTIVIERUNGS_DIALOG.

Über Bedingte Kompilierungskonstanten DIRECT3D, DSOUND und DDRAW werden in den Klassen DIRECT3D, DIRECTSOUND und DIRECTDRAW noch DirectX-Bibliotheken für Direct3D, DirectSound und DirectDraw zur Verfügung gestellt. Des Weiteren stehen in der BASISPROJEKT-Klasse mit WAPICP noch Methoden für den Zugriff auf Sensoren über das Windows API Code Pack zur Verfügung.

Alle Klassen sind in den Namensraum jkBPrj eingeschlossen.

## 6.2 Die BPRJMAINCLASS-Klasse

BPRJMAINCLASS enthält neben der statischen Methode Main, die durch den Compiler als Startobjekt für das Basisprogramm aufgerufen wird, noch zwei Methoden als Handler von globalen Ausnahmen.

Für einen kontrollierten Ablauf wird in der statischen Main-Methode in einem try-catch-Block ein BASISPROJEKT-Objekt mit einer separaten Anweisung erzeugt und die Standardmeldungsschleife für das Basisprogramm in einer zweiten Anweisung mit Application.Run gestartet, wodurch auch die Form sichtbar und bedienbar wird. Nach einer hierin abgefangenen und protokollierten Ausnahme wird das Basisprogramm beendet.

Zum globalen Abfangen und Protokollieren von unbehandelten Ausnahmen werden deshalb zwei Ausnahmehandler erzeugt, nach deren Aktivierung das Basisprogramm fortgeführt werden kann.

**Kodierung 6-1: Die BPRJMAINCLASS-Klasse**

## 6.3 Die BASISPROJEKT-Klasse

Eine Instanz dieser Klasse wird mit new jkBPrj.BASISPROJEKT von Form ableitend erzeugt und stellt hierdurch mit den eigenen Klassen eine grundlegenden Basis für eine Anwendung zur Vererbung bereit.

Die Basisprojektklasse wird in WAPI_Codepack.cs um Methoden zur Erfassung von Sensordaten mit dem Windows API Code Pack erweitert (s. Auf Sensoren zugreifen).

Für einen unterschiedlichen Lizenzschutz-Ablauf sind drei verschiedene Konstruktoren vorhanden.

Diese Klassendefinition muss an erster Stelle des Quellcodes stehen, sonst ist keine Designer-Unterstützung der Form möglich.

**Hinweis 5 zur Designer-Ansicht der Programme!**

Im DEBUG-Ablauf entsteht beim Schließen der Designer-Ansicht für die Anwendungungen durch die Grafikausgabe eine sich wiederholende catch-Ausnahme. Um diese zu verhindern, wird eine Ausnahmemeldung in BASISPROJEKT.Programm nicht in einer Messagebox sondern mit Debug.WriteLine() zum Ausgabefenster ausgegeben.

**Kodierung 6-2: Die BASISPROJEKT-Klasse**

```
#region BASISPROJEKT-Klasse
///*************************************************************************
/// Version 4.0 vom 01.05.18
///*************************************************************************
[CLSCompliantAttribute(true)]
public class BASISPROJEKT : Form
{

    #region Datenfelder und Konstruktoren #endregion

    #region Programmschleife, Kontrollmethode, Titeltext und Thread-Synchronisation #endregion

    #region Basisprogramme, Pause_Methode und Programm-Initialisierung #endregion

    #region StatusBar, FormPanel, Haupt- und Kontext-Menü erzeugen #endregion

    #region Bearbeitung der Form-Ereignisse #endregion

    #region Bearbeitung der Panel-Ereignisse #endregion

    #region Bearbeitung der Menü-Ereignisse #endregion

    #region Speichern und Drucken der BasisBitmap #endregion

    #region Laden von Image-, Audio- und Videodateien #endregion

    #region Sensoren über das Windows API Code Pack einbinden #endregion

}
#endregion
```

## 6.3.1 Die Instanzvariablen

Auf die implizit privaten Instanzvariablen kann nur innerhalb der Basisprojekt-Klasse zugegriffen werden. Die Referenz auf die in den Konstruktoren erzeugte Instanz von BASIS_GLOBALS für einen Zugriff auf die öffentlichen Datenfelder des Basisprojekts aus den abgeleiteten Klassen ist public. Des Weiteren sind die internen Datenfelder der Aufrufkontrolle Systemzeit, Sekunden und DirectoryPath und weitere für informative Lesezugriffe aus der BASIS_PROCS-Klasse public.

**Kodierung 6-3: die Instanzvariablen der BASISPROJEKT-Klasse**

## 6.3.2 Die Konstruktoren

Wird BASISPROJEKT ohne Parameter erzeugt, wird kein Lizenzschutz durchgeführt, weil das öffentliche Datenfeld Lizenz_Ablauf false initialisiert ist.

Der Lizenzschutz von jk-ware wird durchgeführt, wenn der erste Parameter ein string ist. Dieser string wird als Passwort oder als Projektschlüssel verwendet.

Ist der zweite Parameter true, wird ein Lizenzschutz mit Registrierungsschlüssel durchgeführt. Mit einem Zusatzprogramm aus dem CryptoKeyGenerator-Projekt wird hieraus zusammen mit dem Passwort eine Lizenzdatei erstellt. Ist der zweite Parameter false, wird das Programm nur durch eine Passwortabfrage geschützt.

Ist der erste Parameter bool, wird ein erweiterter Lizenzschutz mit dem Licence Protector der Mirage GmbH durchgeführt.

Der Konstruktor wird direkt in Main oder als abgeleitete Basisklasse bei der Erzeugung einer Anwendung aktiviert.

Alle Konstruktoren erstellen eine Instanz von der BASIS_GLOBALS-Klasse für die öffentlich zugänglichen Datenfelder des Basisprojekts und von der BASIS_PROCS-Klasse mit Methoden zur Verwaltung der Ziehleisten und weiteren Hilfsmethoden.

**Kodierung 6-4: Der BASISPROJEKT-Konstruktor für die Einzelversionen**

Für das Multimedia Theater-Projekt darf Lizenz_Ablauf nicht im Konstruktor aktiviert werden, weil sonst der Lizenzschutz für jede gestartete Anwendungsform durchgeführt wird.

**Kodierung 6-5: Der BASISPROJEKT-Konstruktor für die MDI-Version**

**Hinweis 6 zu den Ein- und Ausgabedaten der Kodierungen!**

Wenn die Ein- und Ausgabedaten keiner Klasse zugeordnet sind, handelt es sich um Instanzvariable (Datenfelder) der aktuellen Klasse.

## 6.4 Erweiterungen und Änderungen zur Version 4.0 des Basisprojekts

- Für die automatische Größenänderung von Steuerelementen für HighDpi-Anzeigen wird in der Projektdatei App.config der Zusatz <appSettings><add key="EnableWindowsFormsHighDpiAutoResizing" value="true" /></appSettings zugefügt.
- In BASIS_DIALOG_HelpRequested() wird bei einem fehlenden Hilfetext die Fehlernachricht nicht in einer Messagebox sondern von der HelpProvider-Klasse als Popup-Hilfetext angezeigt. Achtung! Nach der ersten Ausgabe eines Popup-Hilfetextes, erfolgt für die Anzeige von Hilfetexten in Messageboxen keine weitere Botschaft mehr für den Aufruf von BASIS_DIALOG_HelpRequested(). Deshalb werden alle erweiterten Hilfetexte als Popup-Texte angezeigt.
- Für die BASIS_GLOBALS-Struktur Point3D werden die Gleichheits- und Ungleichheitsoperatoren überladen. Hierzu werden auch die Methoden Equals() und GetHashCode() überschrieben.
- Nach dem Programmstart wird in BASISPROJEKT.Programmschleife das globale Datenfeld

Kontrollmethode_Ok true, nachdem die Kontrollmethode durchlaufen wurde und die Menüs erzeugt wurden. Erst wenn Kontrollmethode_Ok true ist, wurde das gobale Datenfeld TESTVERSION mit den Menüs korrekt erzeugt.

- Zum Programmende auf OnClosed kann das Dialogfenster mit den im Programmablauf eingetretenen Ausnahmen unterdrückt werden. Hierzu wird das globale Datenfeld Ausnahmemeldung eingeführt, das auch auf der Basisseite vom Basisdialog ausgewählt werden kann.

- Für das Abspielen von Audio- und Videodateien wird die Bedingte Kompilierungskonstante „DAUDIOVIDEO" eingeführt. Nur bei einer fehlenden Angabe dürfen alle vom Basisprojekt abgeleiteten Programme alle Plattformen der Projektmappe verwenden.
- Wenn eine Multimediadatei angezeigt wird (Multimedia_Anzeige true), werden in BASISPROJEKT.MenueCompleteHandler in der Statusbar der Dateiname und der Tooltiptext dafür angezeigt. Weitere Aufrufe von MenueCompleteHandler erfolgen in ErzeugeFormMenues und der Pause-Property.
- Für den Wiederaufbau von Multimedia-Theater direkt nach dem Programmstart, werden der Dateiname in DialogFileName und das globale Datenfeld Multimedia_Anzeige in BASISPROJEKT.Programm_Init gespeichert und wieder zurückgeladen. Wegen der undefinierten Größe des Dateinamens wird hierzu die Konstante BASISDATENENDE auf 200 erhöht.
- Das Abspielen von Audio- und Videodateien wird mit der AudioVideoPlayBack-API von Managed DirectX realisiert. Über den „Laden"-Menüpunkt des Basiprogramms in BASISPROJEKT. Image_Audio_Video_Laden werden Multimediadateien der aufrufenden Programmform zugeordnet und abgespielt. Eine weitergehende Steuerung der horizontalen Ziehleiste erfolgt mit der Tastatur in OnKeyDown und in Rollbalkenverw und HorzRollbalken.
- Der Positionsabgleich der Audio- und Videodatei mit der horizontalen Ziehleiste erfolgt in PanelOnPaint. Hierin wird auch ein Bild oder Video an die Panelgröße angepasst, wenn es nicht in Originalgröße angezeigt werden soll.
- Für Multimediadateien wird zur Aktualisierung der horizontalen Ziehleiste und der Paneltexte der Statusleiste ein Basiszeitgeber in BASISPROJEKT.OnLoad eingerichtet und in Image_Audio_Video_Laden gestartet. In der Zeitgebermethode Basis_Zeitgeber wird die horizontale Ziehleiste mit der Position des Videos oder Audios aktualisiert. Des Weiteren werden die erforderlichen Panels verwaltet und ihnen ein Text und Tooltiptext zugewiesen.
- Das Basismenü wird um den Menüpunkt „Originalgröße" erweitert. Wenn dieser markiert ist, wird ein Bild in der Originalgröße angezeigt und ein Video in der Ersatzgröße abgespielt.
- In BASISPROJEKT.Basis wird die Aufrufkontrolle nur durchlaufen, wenn keine Multimediadatei angezeigt wird.

- Die Anzeige von Multimediadateien wird auch mit Drag und Drop realisiert. Hierzu werden in ErzeugeFormPanel() für die DragEnter- und DragDrop-Ereignisse die erforderlichen Handler PanelOnDragEnter und PanelOnDragDrop eingerichtet (s. a. Literatur 2. Petzold, Kapitel 24).

### 6.4.1 Codeanalyse mit Visual Studio Community

Von der Codeanalyse für den ausgeführten Regelsatz „Alle Microsoft Regeln" werden viele Warnungen angezeigt, die bisher noch nicht alle beseitigt wurden.

Der ausgeführte Regelsatz kann unter den Projekteigenschaften\Codeanalyse ausgewählt werden. Von dem Regelsatz „Alle Microsoft Regeln" ausgehend, wird über den Öffnen-Taster durch Visual Studio eine neue Seite mit einer Liste von Warnungen angezeigt, die über Checkmarken abgewählt werden können.

Danach steht für das Basisprojekt ein eigener Regelsatz zur Auswahl:

- Daraus wurden wegen Irrelevanz folgende Warnungen abgewählt:
  CA1021: out-Parameter vermeiden
  CA1045: Methoden keine Verweise übergeben (keine ref-Typen)
  CA1062: Argumente von öffentlichen Methoden vor der Verwendung überprüfen
  CA1300: MessageBoxOptions angeben für MessageBox.Show
  CA1303: Texte von einer Resourcentabelle abrufen
  CA1304: IFormatProvider angeben für Methodenaufrufe wie string.Compare
  CA1305: IFormatProvider angeben für Methodenaufrufe wie string.Format
  CA1307: Zeichenfolgen mit StringComparison.Ordinal-Parameter vergleichen
  CA1502: übermäßig komplexe Methoden vermeiden
  CA1505: nicht wartbaren Code vermeiden (z.B. für globale TEXTE-Klassen)
  CA1506: Übermäßige Klassenkopplungen vermeiden
  CA1601: verwenden Sie keine Zeitgeber, die häufig ein Ereignis auslösen (< 1 s)
  CA1704: aussagekräftige Bezeichner verwenden
  CA1707: Bezeichner sollten keine Unterstriche enthalten
  CA1709: bei Bezeichnern die Groß-/Kleinschreibung beachten
  CA1801: Eine Methodensignatur enthält einen nicht verwendeten Parameter
  CA1811: nicht aufgerufenen privaten Code vermeiden
  CA1814: Verzweigte Arrays mehrdimensionalen Arrays vorziehen (z.B für Texte)
  CA1822: Methoden, die nicht auf Instanzdaten zugreifen, als statisch markieren
  CA1823: nicht verwendete private Felder vermeiden
  CA1824: Assemblies mit NeutralResourcesLanguageAttribute markieren
  CA2211: ein statisches Feld ist weder konstant noch schreibgeschützt (z.B Text-Arrays)

- Folgende Warnungen wurden wegen zu großen Aufwandes abgewählt:
  CA1031: keine allgemeinen Ausnahmen auffangen
  CA1034: keine sichtbaren geschachtelten Typen
  CA1051: keine sichtbaren Instanzfelder deklarieren (keine globalen Daten)
  CA1053: statische Klassen definieren, wenn diese nur statische Daten enthalten
  CA2000: Objekte verwerfen, bevor der gültige Bereich verloren geht (betrifft auch globale Objekte, die im gesamten Programmablauf benötigt werden)
  IDE0017: die Objektinitialisierung kann vereinfacht werden (mit Klammern)

- Folgende Warnungen wurden behoben:
  CS0168: deklarierte Variable ist ohne Verwendung
  CA1011: Basistypen als Parameter verwenden:
  In Managed_DirectX.cs z.B. für DeviceDrawTexture (BaseTexture texture) anstelle von Texture als Parametertyp übergeben.
  CA1017: Assembly-Informationen ohne Checkmarke „Assembly Com-sichtbar machen"
  CA1026: keine Parameter initialisieren für Point3D und PointF3D
  CA1500: Parameternamen ändern, wenn ein Instanzenfeld denselben Namen hat.
  CA1702: bei zusammengesetzten Begriffen die Groß-/Kleinschreibung beachten
  CA1720: Bezeichner dürfen keine Typnamen enthalten (z. B. nicht obj)
  CA1725: EventArgs ea zu EventArgs e geändert
  CA1820: String.IsNullOrEmpty(string) verwenden
  CA2002: Objekte nicht mehrmals verwerfen
  CA2201: eine Methode löst einen Ausnahmetyp aus, der zu allgemein ist
  CA2210: Assemblys müssen gültige starke Namen aufweisen:
  anhand der Fehlerbeschreibung wurde die Schlüsseldatei Basisprojekt.snk erstellt.
  CA2233: Arithmetische Berechnungen auf einen möglichen Überlauf prüfen

## 6.4.2 Für neue Versionen vorgesehen

- Die Main-Methode der CryptoKeyGenerator-Klasse für einen externen Server-Aufruf mit Argumenten für den Registrierungsschlüssel und das Passwort versehen (s. 7. Programmierrezepte für Visual C# .Net, S. 436f und 8. C# IT-Tutorial, S. 264f).

### 6.4.3 Erweiterungen und Änderungen zur Version 3.0 des Basisprojekts

- Das Zielframework wird für alle Projekte auf .Net Framework 4 Client Profile eingestellt. Hiermit wird auf die neuen Betriebssysteme ab Windows 7 eingegangen, bei denen nur noch diese Basis installiert ist.
- Einführung der DirectX-Bibliothek 1.0 zur allgemeinen Verwendung durch die abgeleiteten Projekte in Managed_DirectX.cs.
- Zugriff auf Licht- und Beschleunigungssensoren mit Methoden des Windows API Code Pack in WAPI_CodePack.cs.
- In Basis() wird in Aufrufkontrolle() AufrufFilterAnz nicht bei einer Spielpause neu berechnet.
- Kontrollmethode() gibt einen Rückgabewert zurück.
- Die Menüs werden in Programmschleife(), nach dem Aufruf von Kontrollmethode() neu erzeugt und nicht mehr in Kontrollmethode(), weil diese auch innerhalb des Programmablaufs aufgerufen wird und es hierbei zu Deadlocks kommen kann.
- Zum Sperren einzelner Menüpunkte werden für das Programm-Menü und das Kontextmenü Ereignisse für das Popup-Ereignis erzeugt. Zum Überschreiben durch die Anwendung werden für diese Ereignisse virtuelle Handler ohne Inhalt zur Verfügung gestellt.
- Der Zugriffsmodifikator für das Feld DirectoryPath und die Methoden Programm, Titeltext und InitializeComponent wird von public zu protected geändert und das Feld BasisProcs wird private. Das Feld BGlobals und die Methode Programm_Init müssen für die MDI-Projekte public bleiben.
- Definition allgemeiner öffentlicher Strukturen Point3D, PointF und PointF3D in Basis_Globals.cs für einen Zugriff im ganzen Projekt.
- Neue virtuelle und inhaltslose Botschaftsmethode FormOnMove für das Move-Ereignis der Form zum Überschreiben durch die Anwendungen (das Move-Ereignis der Panels erzeugt keine Botschaften).
- In Programm_Init() werden wegen einer möglichen Ausnahme die Instanzen von BReader und BWriter innerhalb eines finally-blocks geschlossen.

### 6.4.4 Erweiterungen und Änderungen zur Version 2.0 des Basisprojekts

- Zur weitergehenden Anzeige im Objektkatalog und für die kontextsensitive Hilfe wurde der XML-Kommentar für alle Methoden mit <para>-Tags innerhalb der <summary>- und <remarks>-Tags versehen.
- Neben der Erhöhung der globalen Variablen catch_Ausnahmen, werden die innerhalb des Programmablaufes auftretenden Ausnahmen mit Datum und Uhrzeit in Ausnahmeprotokollierung der BASISPROJEKT-Klasse in eine Datei geschrieben. Hierzu wird im Basisprogramm auf OnLoad ein Ausgabestream für die Datei „Titeltext" + "_Exceptions.txt" erzeugt, der auf OnClosed wieder geschlossen wird (für das MDI-Projekt heißt die Ausnahmedatei für alle als Formen gestarteten Programme immer „MDIFormen_Exceptions.txt").
- Zum globalen Abfangen und Protokollieren von unbehandelten Ausnahmen werden in BPRJMAINCLASS.Main zwei Ausnahmehandler erzeugt, nach deren Aktivierung die Anwendung fortgeführt wird.
- Separate Kontrollmethode für den Lizenzschutz. Der Aufruf erfolgt nach dem Programmstart in Programmschleife und zur zusätzlichen Lizenzkontrolle in dem Menüereigniss OnMenueComplete.
- Größenänderungen des Panels werden durch den in ErzeugeFormPanel() erzeugten virtuellen EventHandler PanelOnResize bearbeitet. Hierin werden auch das GDI+ Graphics-Objekt und die

Basisbitmap erzeugt. Eine fehlerhafte Panel-Erzeugung in ErzeugeFormPanel wird nicht abgefangen und führt zum Programmabbruch (bisher noch nicht vorgekommen).

- Des Weiteren werden in ErzeugeFormPanel für das Panel vertikale und horizontale ScrollProperties und für das Scroll-Ereignis der virtuelle ScrollEventHandler PanelOnScroll erzeugt. Für die MouseDown- und MouseMove-Ereignisse werden die EventHandler PanelOnMouseDown und PanelOnMouseMove virtuell zum Überschreiben durch die Anwendung erzeugt.
- Die Methoden OnPaint und OnMouseDown zur Bearbeitung der Form-Ereignisse Paint und MouseDown sind nicht mehr erforderlich und werden entfernt. Achtung! Zum Neuzeichnen der Form in PanelOnPaint muss anstelle von Invalidate, FormPanel.Invalidate verwendet werden.
- Zur Bearbeitung der Ereignisse von den Ziehleisten auf OnScroll werden in der BASIS_PROCS-Klasse die Methoden VertRollbalken und HorzRollBalken zur Verfügung gestellt. Hierdurch werden Probleme mit einer Anwendung ohne AutoScroll nach einer Bildanzeige mit AutoScroll vermieden.
- Die Methode Rollbalkenverw der BASIS_PROCS-Klasse wird zur grundlegenden Einstellung der Positionsgrenzen und Abgriffpositionen der horizontalen und vertikalen Ziehleisten für eine in der Form angezeigte Bitmap eingeführt.
- Für eine angezeigte Bilddatei, werden die Ziehleisten in BASISPROJEKT.OnKeyDown mit den Pfeil- und Sondertasten der Tastatur bewegt. Wenn keine Ziehleiste angezeigt ist, wird dadurch der Cursor bewegt.
- Eine Bewegung des Mausrades wird durch den überschreibenden Handler für das MouseWheel-Ereignis **der Form** in OnMouseWheel in eine Bewegung der Ziehleisten umgesetzt.
- Neue BASIS_GLOBALS-Instanzvariable Pause_Aussetzen. Wenn true, wird in der Basis-Methode der Pausestatus des Programms, z.B. zur Aktualisierung der Grafikanzeige, vorübergehend ausgesetzt ohne dass dabei die Pause-Menücheckmarken verändert werden.
- Die Felder Pause und StBarAnzeige der BASIS_GLOBALS-Klasse werden als Eigenschaften definiert, die zusätzlich die Checkmarken des Haupt- und Kontextmenüs mit verändern. Pause wird auch in Programm_Init als Initialisierungsdatum abgespeichert.
- Die BASIS_GLOBALS-Instanzvariable DirectoryPath entfällt, da sie dem Application.StartupPath entspricht.
- Zum Beenden einer Bildanzeige wird für den Zugriff auf lokale Daten und Methoden wird die Pause_Methode eingeführt, in der die Menücheckmarken korrigiert und die Ziehleisten verdeckt werden. Zur Anpassung der Panelgröße wird PanelOnResize() aufgerufen. Der Aufruf erfolgt beim Setzen der Eigenschaft für die globale Instanzvariable Pause.
- In OnResize wird bei einer Verkleinerung der Form zur Symbolgröße Pause nicht true gesetzt, weil hierdurch über die neue Pause_Methode eine Bildanzeige beendet wird.
- Die Instanzvariable SymbolSize der BASISPROJEKT-Klasse wird zur Instanzvariablen der BASIS_GLOBALS-Klasse.
- Ausgabe der Bitzahl der .Net-Platform in der über-Dialogbox des Menüs und der Basis-Dialogseite. Die Bitanzahl wird von der Länge eines int-Pointers abgeleitet.
- Für einen schnelleren Programmablauf in der Programmschleife wird die max. Anzahl der Ereignisabfragen von 100 auf 50 reduziert (nicht für MULTITHREAD).
- In ThreadSynchroStart wird die Synchronisationsschleife mit dem Arbeitsthread bei einer fehlenden Synchronisation anstelle von SynchroStatus = true mit einer selbst erzeugten Ausnahmeinstanz beendet. Hierdurch ist auch nachfolgend noch ein korrekter Synchronisationsablauf möglich. Des Weiteren wurden die Werte der Sleep()-Anweisungen verändert, um einen Threadwechsel zu erzwingen oder die Wartezeit zu verringern.
- Um einen Threadwechsel zu erzwingen werden in MT_Programmschleife() die beiden Sleep(0)-Anweisungen durch Sleep(1)-Anweisungen ersetzt.
- Zum längeren pausieren des Arbeitsthread wird die Methode Arbeitsthread_pausieren() eingeführt. In der Pausezeit wird zur Vermeidung eines Deadlocks ein Wechsel zum Bedienerthread ermöglicht.

- Aufgrund von Problemen mit der Mausposition als Datenelement von HelpEventArgs zur Anzeige der erweiteren Hilfetexte in BASIS_DIALOG::BASIS_DIALOG_DIALOG_HelpRequested und ANW1_DIALOG::Anw1_DIALOG_HelpRequested, wird die aktuelle Mausposition von der Control-Klasse verwendet.
- Die Druckerausgabe in BASISPROJEKT::BasisBitmapDrucken wird über Anzeigen der PageSetup- und darauf folgender PrintPreview-Dialogform durchgeführt, wenn die Ausgabe mit dem Standard Drucker-Dialog von der System.Windows.Forms.PrintDialog-Klasse nicht realisiert werden kann oder abgebrochen wird.
- In LIZENZSCHUTZ::Kontrollablauf wird für die Passwortabfrage in der binären Kontrolldatei der Kontrolltext aus dem Programmpfad und der Erstellungszeit des Programmordners gebildet.
- In LIZENZSCHUTZ::ProtectMe von der mirage GmbH wird der Lizenzschutz in der Version 3.1 ausgeführt.
- Die multilingualen Lizenztexte („EULA“) für den Lizenzdialog in der LIZENZSCHUTZ-Klasse in Basis_Procs.cs auf den 05.06.2009 korrigiert. Hierbei nur drei kleine grammatikalische Korrekturen an dem deutschen und englischen Text vorgenommen. Der Erläuterungstext für die Registrierungsschlüssel-Dialogform wurde verändert.
- Die globale Instanzvariable TESTVERSION wird in Programm_Init als Init-Datum abgespeichert und nicht für das MDI-Projekt eingelesen.

### 6.4.5 Änderungen zur Version 1.1 des Basisprojekts

- Im Formenverwaltungs-Programm hat die Kindform eine unkorrekte Panelgröße, wenn das Programm mit maximierter Form gestartet wird. Zur Korrektur wird in OnResize deshalb die Panelgröße explizit berechnet.
- am Ende von OnLoad wird OnResize explizit aufgerufen. Dies ist erforderlich, wenn kein Menü angezeigt wird, da sonst kein Grafik-Objekt für das Panel erzeugt wird.
- Neuer Ablauf zur Threadsynchronisation in MT_Programmschleife, ThreadSynchroStart und ThreadSynchroEnde. ThreadSynchroStart setzt zur Anforderung einer Synchronisation mit dem Arbeitsthread die Variable ThreadSynchro true. Die Pause-Variable der Anwendung wird während des Synchronisationsablaufes nicht mehr verändert.

## 6.5 Die öffentlichen Datenfelder des Basisprojekts

Datenfelder, die für andere Klassen öffentlich zugänglich sein müssen, werden in einer eigenen Klasse BASIS_GLOBALS aufgeführt. Auf diese Datenfelder kann hierdurch über eine separate Referenz zugegriffen werden, die auch austauschbar ist. Die BASIS_GLOBALS-Klasse wird in den Konstruktoren des Basisprojekts erzeugt.

Im Konstruktor werden die öffentlichen Datenfelder des Basisprojekts initialisiert und ein Generator für Pseudozufallszahlen erzeugt.

Von BASIS_GLOBALS werden automatisch Objekte von BASIS_TEXTE und BASIS_MENUE_TEXTE als Basisklassen abgeleitet. Diese Enthalten statische zwei- oder dreidimensionale Arrays in denen die mehrsprachigen Texte für das Basisprojekt und das Menü eingetragen werden.

Die Menütexte enthalten zusätzlich einen kleinen Erläuterungstext zur Menüauswahl, der für die Ausgabe zur Statusleiste verwendet wird.

**Kodierung 6-6: die BASIS_GLOBALS-Klasse**

### 6.5.1 Mehrsprachigkeit

Die mehrsprachige Bedienung der Programme wird über statische zweidimensionale string-Arrays realisiert, die als öffentliche Datenfelder zugänglich sind. Statische Arrays haben den Vorteil, dass sie auch bei mehreren erzeugten Instanzen nur einmal im Speicher vorhanden sind.

Beispiele:

Eine Messagebox mit einer mehrsprachigen Nachricht wird mit MessageBox.Show (BASIS_TEXTE.MessageBoxTexte[User_Language-1,1], Text) angezeigt. Hierbei ist MessageBox.Show eine Methode der System.Windows.Forms-Klasse. Text übergibt den Titeltext der Form. BASIS_TEXTE.MessageBoxtexte[,] wählt ein Element eines zweidimensionalen statischen Arrays aus. User_Language-1 ist die ausgewählte Textsprache und erst die darauffolgende eins gibt den Text für die Messagebox an.

Eine mehrsprachige Textausgabe zum Panel1 der Statusleiste erfolgt in dieser Art und Weise mit der Anweisung StBarPanel1.Text = BASIS_TEXTE.BasisTexte [BGlobals.User_Language-1,2]. Mit BGlobals als Instanz der öffentlichen Datenfelder, die außerhalb der BASISPROJEKT-Klasse für den Zugriff auf die Texte erforderlich ist.

### 6.5.2 Die mehrsprachigen Basistexte

Die BASIS_TEXTE-Klasse enthält allgemein zugängliche multilinguale Texte für das Basisprojekt und erbt BASIS_MENUE_TEXTE.

Im Konstruktor werden die allgemeinen multilingualen Texte für das Basisprojekt in den Textarrays bereitgestellt.

Zur Seiteneinsparung werden nur die deutschen und englischen Texte aufgeführt.

**Kodierung 6-7: Die BASIS_TEXTE-Klasse**

### 6.5.3 Die multilingualen Texte des Basismenüs

Die BASIS_MENUE_TEXTE-Klasse enthält allgemein zugängliche multilinguale Texte für das Basismenü mit einem kleinen Erläuterungstext zur Menüauswahl. Zur Seiteneinsparung werden nur die deutschen und englischen Texte aufgeführt.

**Kodierung 6-8: Die BASIS_MENUE_TEXTE-Klasse**

## 6.6 Das Panel

Für eine korrekte Gruppierung der Ziehleisten und der Statusleiste innerhalb der Form wird eine Panel-Kontrolle erzeugt. Hierdurch kann die Grafikfläche mit Ziehleisten ohne die Statusleiste verschoben werden.

Eine unkorrekte Erzeugung der Panel-Instanz führt zum Abbruch der Anwendung (bisher noch nicht vorgekommen).

**Hinweis 7 zu den Ziehleisten!**

Weil die Statusleiste zur Grafikfläche der Form gehört, wird diese mit verschoben, wenn die Form mit Ziehleisten erzeugt wird. Hierbei handelt es sich noch um eine grundlegende Eigenart der Windows-Programmierung mit leider aufwendigen Folgen für den Programmierer.

Zur Behandlung der Panel-Ereignisse Resize, Paint, MouseDown, MouseMove, Scroll, DragEnter und DragDrop werden die Methoden PanelOnResize, PanelOnPaint, PanelOnMouseDown,

PanelOnMouseMove, PanelOnScroll, PanelOnDragEnter und PanelOnDragDrop als Handler eingetragen.

Für das Move-Ereignis der Form wird die Ereignismethode FormOnMove als Handler eingetragen, weil das Panel selbst kein Move-Ereignis erzeugt.

Der Aufruf erfolgt im Form-Ereignis OnLoad.

**Kodierung 6-9: ErzeugeFormPanel**

## 6.7 Die Statusleiste

Zur Anzeige von Programm- und Multimedia-Informationen wird eine Statusleiste mit drei Panel erzeugt.

Dem ersten Panel wird ein Text und übergeordnet ein ToolTipText zugewiesen. Die beiden weiteren Panel werden nur erzeugt aber nicht der PanelCollection der Statusbar zugefügt.

Die Statusleiste wird nur angezeigt, wenn das öffentliche Datenfeld StBarAnzeige true ist.

Der Aufruf erfolgt im Form-Ereignis OnLoad.

**Kodierung 6-10: ErzeugeStatusbar**

## 6.8 Der Titeltext

Setzt den Text in die Titelleiste der Form des Basisprogramms oder einer überschreibenden Anwendung zum übergebenen string-parameter.

Nur für das MDI-Projekt wird die Programmnummer mit Punkt dem Programmtitel vorangesetzt.

Der Aufruf erfolgt im Basisprojekt auf OnLoad und in ErzeugeFormMenues.

**Kodierung 6-11: Titeltext für die Form**

Eine überladene Methode von Titeltext setzt den Text der Titelleiste der Form zum übergebenden string-Parameter.

Hierin wird nur für MULTITHREAD die Ausführung über einen neu erzeugten Delegaten auf den Bediener-Thread umgeleitet, wenn der Aufruf nicht aus dem Bediener-Thread erfolgte und somit InvokeRequired true zurückgibt.

**Kodierung 6-12: Titeltext für die Form mit string-Parameter**

## 6.9 Die Grafikobjekte

Das Basisprojekt stellt zwei Grafikobjekte für die Ausgabe zur Grafikfläche (Client Area) des Panels und zur Ausgabe in eine Basisbitmap in der Größe der Grafikfläche zur Verfügung.

Die Anwendung kann über diese Grafikobjekte entweder direkt auf dem Display zeichnen oder alles zuvor in eine Basisbitmap ablegen, um diese dann insgesamt zur Client Area zu übertragen.

Die Grafikobjekte werden bei einer neuen Panelgröße in der in ErzeugeFormPanel für das Resize-Ereignis des Panels eingetragenen Methode PanelOnResize neu erzeugt.

Ist die zu zeichnende Grafik größer als die Grafikfläche der Form, muss die Basisbitmap mit dem davon abgeleiteten Grafikobjekt von der Anwendung selbst erzeugt werden. Dazu muss die öffentliche Variable Grafik_Anzeige true sein, so dass in PanelOnResize keine Basisbitmap mehr erzeugt wird.

## 6.10 Bearbeitung der Form-Ereignisse für das Basisprogramm

### 6.10.1 OnLoad

In OnLoad werden das Haupt- und Kontextmenü, eine Statusbar, eine Panel-Kontrolle über der Form, ein Sekunden Zeitgeber und die Instanz zum Schreiben der Fehlermeldungen in eine Datei erzeugt.

Des Weiteren werden die Startdaten für das Programm aus der Initialisierungsdatei eingelesen, das Symbol für das Basisprogramm aus den Ressourcen geladen und Handler für das MenuStart- und MenuComplete-Ereignis eingetragen.

Wenn Multimedia_Anzeige true ist, wird die zuletzt angezeigte Multimediadatei wieder geladen (wenn noch vorhanden).

Nur für MULTITHREAD wird ein Arbeitsthread mit MT_Programmschleife als Startmethode und nicht für das MDI_PRJ ein Zeitgeber zur Aktivierung von Programmschleife gestartet.

Anm.: In Programmscheife wird für den Ablauf mit MULTITHREAD nur ein erforderlicher Kontrollablauf durchgeführt.

OnLoad überschreibt die geerbte Form-Methode und wird nur einmal aufgerufen, bevor die Form angezeigt wird.

**Kodierung 6-13: OnLoad für das Basisprogramm**

### 6.10.2 OnResize

OnResize überschreibt die geerbte Form-Methode und durchläuft, wenn die Form nicht zum Symbol verkleinert oder wiederhergestellt wird, zur Aktivierung der eingetragenen Resize-Handler die Basismethode.

OnResize wird bei einer Größenänderung der Form vor dem Resize-Ereignis aufgerufen.

**Kodierung 6-14:  OnResize für das Basisprogramm**

### 6.10.3 FormOnMove

FormOnMove ist der in ErzeugeFormPanel eingetragene Handler für das Move-Ereignis der Form und wird virtuell und inhaltslos zum Überschreiben durch das Programm definiert.

**Kodierung 6-15: FormOnMove für das Basisprogramm**

### 6.10.4 OnMouseWheel

OnMouseWheel ist eine überschreibbare Methode zur Verwaltung der Ziehleisten nach einer Bewegung des Mausrades über der Form.

Für eine angezeigte Bilddatei gilt, wenn die Bewegung des Mausrades in der oberen Hälfte der Form erfolgt, wird die Stellung des vertikalen Rollbalkens geändert. Wird das Mausrad in der unteren Formhälfte bewegt, beeinflusst dies die Stellung der horizontalen Ziehleiste.

Bei einer abgespielten Video- oder Audiodatei wird über der ganzen Form nur die Stellung der horizontalen Ziehleiste beeinflusst (nur für DAUDIOVIDEO).

OnMouseWheel überschreibt die Basismethode der Form und wird bei einem über der Form gedrehten Mausrad aufgerufen.

**Kodierung 6-16: OnMouseWheel für das Basisprogramm**

## 6.10.5 OnKeyDown

Nach einer gedrückten Pause-Taste wird die öffentliche Eigenschaft Pause true, wodurch auch die Pause-Checkmarken beider Menüs gesetzt werden.

Auf betätigte Pfeiltasten, auch der Zehnertastatur, wird bei einer angezeigten Ziehleiste, die Ziehleiste, sonst der Mauszeiger weiter bewegt (mit gedrückter Strg-Taste um 5 Pixel).

Nach Betätigung der Pos 1-, Ende-, Bild-Auf- und Bild-Ab-Tasten werden die Ziehleisten entsprechend positioniert.

Bei der Bewegung der Ziehleisten wird die Ober- und Untergrenze des bildlauffähigen Bereiches beachtet.

Bei der Wiedergabe von Video- und Audiodateien werden nach Betätigung der Media Play- und Pause-Taste und der Pause-Taste das Video und der Klang gestoppt und wieder gestartet und die Werte der horizontalen Ziehleiste angepasst.

OnKeyDown überschreibt die geerbte Form-Methode und wird bei jedem Tastendruck aufgerufen, wenn die Taste nicht als Shortcut-Taste für ein Menüelement verwendet wird.

**Kodierung 6-17: OnKeyDown für das Basisprogramm**

## 6.10.6 OnClosed

OnClosed schreibt in Programm_Init wichtige Programmdaten für den nächsten Programmstart in die Datei unter Init_Dateiname.

Nachdem die Schleifenvariable von der Programmschleife false gesetzt wurde, wird für MULTITHREAD auf die Beendigung des Arbeitsthreads gewartet, in dem die Programmschleife läuft.

Wenn während des Programmablaufes innerhalb der try-Blöcke catch-Ausnahmen abgefangen wurden, wird die Anzahl in einer Messagebox angezeigt und auf die Ausnahmeprotokolldatei verwiesen.

Erst danach werden noch vorhandene öffentliche Objekte explizit freigegeben, weil diese noch für die Anzeige der Messagebox benötigt werden.

OnClosed überschreibt die von der Form geerbte Methode und wird einmal aufgerufen, nach dem die Form geschlossen wurde.

**Kodierung 6-18: OnClosed für das Basisprogramm**

### 6.10.7 Dispose

Die von Form verwendeten Ressourcen (mit Ausnahme des Speichers) werden freigegeben.

Dispose überschreibt die geerbte Form-Methode und ruft diese selbst als Basismethode auf. Der Aufruf erfolgt nach OnClosed, nachdem die Form geschlossen wurde.

**Kodierung 6-19: Dispose für das Basisprogramm**

## 6.11 Bearbeitung der Panel-Ereignisse für das Basisprogramm

### 6.11.1 PanelOnResize

PanelOnResize erzeugt zur Ausgabe direkt zum Display zur allgemeinen Verwendung ein GDI+ Grafik-Objekt, das in der neuen Größe des Panels erstellt wird. In dieser Größe wird auch eine Basisbitmap mit einem davon abgeleiteten GDI+ Grafik-Objekt erstellt.

Die Basisbitmap wird aber nicht erstellt, wenn sie extern zur Verfügung gestellt wird (Grafik_Anzeige true) oder eine Multimediadatei angezeigt wird (Multimedia_Anzeige true). In beiden Fällen wird PanelOnResize vorzeitig verlassen. Vorher werden die Ziehleisten in Rollbalkenverw auf die neue Formgröße eingestellt.

Für die erzeugten Grafik-Objekte, wird die Grafikqualität und der Pixeloffset entsprechend der Eingabedaten gesetzt.

Für MULTITHREAD wird vor der Behandlung der Grafik-Objekte der Bedienerthread mit dem Arbeitsthread synchronisiert.

PanelOnResize wird bei einer Größenänderung des Panels als eingetragener Handler für das Resize-Ereignis aufgerufen.

**Kodierung 6-20: PanelOnResize für das Basisprogramm**

### 6.11.2 PanelOnPaint

PanelOnPaint aktualisiert die Video- oder Audioposition mit dem Wert der horizontalen Ziehleiste.

Wenn das Video nicht in der Originalgröße angezeigt werden soll, wird es von der Ersatzgröße zur Panelgröße umgerechnet.

Bei der Anzeige der Basisbitmap wird die horizontale und vertikale Position der Ziehleisten berücksichtigt, wenn die Bitmap größer als das Panel sein kann (nicht Multimedia_Anzeige und Grafik_Anzeige oder Image_Anzeige und Originalgroesse).

Für eine Bildanzeige nicht in Originalgröße (keine Animation), wird das in der Basisbitmap an die Panelgröße angepaßte Bild zum Display kopiert.

Die Basisbitmap wird nicht bei einem Videoablauf ausgegeben.

Für MULTITHREAD wird vor dem Zugriff auf das Grafik-Objekt der Bedienerthread mit dem Arbeitsthread synchronisiert.

PanelOnPaint ist der Handler für das Paint-Ereignis des Panels und wird aufgerufen, wenn die visuelle Darstellung des Panels erneuert werden muss.

**Kodierung 6-21: PanelOnPaint für das Basisprogramm**

### 6.11.3 PanelOnMouseDown

Nach einer auf dem Panel gedrückten rechten Maustaste wird die öffentliche Eigenschaft Pause true wodurch auch die Pause-Checkmarken beider Menüs gesetzt werden.

Der Aufruf von PanelOnMouseDown erfolgt als eingetragener Handler für das MouseDown-Ereignis des Panels nach einer gedrückten Maustaste.

**Kodierung 6-22: PanelOnMouseDown für das Basisprogramm**

### 6.11.4 PanelOnMouseMove

PanelOnMouseMove ist der in ErzeugeFormPanel eingetragene Handler für das MouseMove-Ereignis des Panels und wird virtuell und inhaltslos zum Überschreiben durch das Programm definiert.

**Kodierung 6-23: PanelOnMouseMove für das Basisprogramm**

### 6.11.5 PanelOnScroll

PanelOnScroll ist eine virtuelle Methode zur Verwaltung der Ziehleisten des Panels bei einer Bildanzeige.

Für die Bearbeitung der Bewegungen der vertikalen Ziehleiste wird VertRollbalken und für die Veränderungen der horizontalen Ziehleiste HorzRollbalken der BASIS_PROCS-Klasse aufgerufen.

Der Aufruf erfolgt als ein in ErzeugeFormPanel eingetragener Handler für Scroll-Ereignisse der Ziehleisten des Panels.

**Kodierung 6-24 : PanelOnScroll für das Basisprogramm**

### 6.11.6 PanelOnDragEnter

PanelOnDragEnter überprüft, ob die über das Panel gezogenen Daten angezeigt werden können. Zur Kontrolle wird hierzu Multimediadatei_OK aufgerufen.

Wenn es sich um eine anzeigbare Multimediadatei handelt, wird der DragDrop-Cursor zu Copy verändert (nicht für die Testversion).

Der Aufruf erfolgt als ein in ErzeugeFormPanel eingetragener Handler für das DragEnter-Ereignis des Panels.

**Kodierung 6-25 : PanelOnDragEnter für das Basisprogramm**

### 6.11.7 PanelOnDragDrop

PanelOnDragDrop zeigt die über das Panel gezogene Datei an.

Hierzu wird in Image_Audio_Video_Laden() der von Drag und Drop hinterlegte Dateiname angezeigt.

Der Aufruf erfolgt als ein in ErzeugeFormPanel eingetragener Handler für das DragDrop-Ereignis des Panels, wenn in PanelOnDragEnter der DragDrop-Effekt auf Copy gesetzt wurde.

**Kodierung 6-26: PanelOnDragDrop für das Basisprogramm**

# 7 Der Basisprogrammablauf

## 7.1 Die Initialisierung des Basisprogramms

In Programm_Init werden relevante öffentliche Datenfelder des Basisprojekts in eine für das Schreiben neu erzeugten Datei unter Filename geschrieben oder aus einer bestehenden Datei eingelesen.

Nach dem Lesen der Datenfelder ist in dem öffentlichen Datenfeld FileStreamPos die aktuelle Streamposition abgelegt.

**Hinweis 8 zur Erweiterung der Init-Daten in neuen Versionen des Basisprogramms!**

Für einen fehlerfreien Zugriff auf neu zugefügte Datenfelder in den **alten** Init-Dateien, wird das Datenende auf die konstante BASISDATENENDE vorsorglich nach oben verschoben. Hierbei ist zu beachten das eine string-Erweiterung nur einen Positionswert aus den alten Init-Dateien beansprucht und der eingelesene „string" somit fehlerhaft ist. Integerwerte erhalten auch erstmalig einen falschen Wert, werden aber mit einer korrekten Positionsanzahl eingelesen.

Nur für das MDI-Projekt wird zum übergebenen Dateinamen vor der Erweiterung die Programmnummer aus dem öffentliche Datenelement ProgrammNr zugefügt.

Der Aufruf zum Lesen der Datenfelder erfolgt nach dem Programmstart in OnLoad. Der Aufruf zum Schreiben erfolgt zum Programmende auf OnClosed.

Programm_Init ist eine virtuelle Methode, die von der Anwendung zur Abspeicherung eigener Datenfelder überschrieben werden kann.

**Kodierung 7-1: Programm_Init für das Basisprogramm**

## 7.2 Die Programmschleife (nicht für MDI_PRJ)

Nur für die Einzelversionen wird diese Programmschleife als Ereignishandler eines in OnLoad erzeugten Timers durchlaufen. Der Ablauf erfolgt bis zum Verlassen der Schleife mit ProgLoop false.

Zu Beginn wird der aufrufende Timer gestoppt und die Kontrollmethode für die Lizenzkontrolle ausgeführt.

Wenn der Kontrollablauf korrekt abläuft, wird die öffentliche Variable TESTVERSION false und das Programm läuft in der Endversion.

Zur Anpassung der Menüs an die Bedienersprache werden diese nach dem Kontrollablauf neu erzeugt. Hierbei wird auch der Text der Titelleiste aktualisiert.

Für MULTITHREAD erfolgt nach der Lizenzkontrolle der Rücksprung aus der Methode, ohne in der Programmschleife zu verbleiben.

Nur für die Einzelthread-Versionen wird darauf in einer Programmschleife ständig Basis aufgerufen und die Programm-Ereignisse abgearbeitet. Zur weitergehenden Optimierung kann die Anzahl dieser Ereignisabfragen begrenzt werden (Standardwert 50).

**Kodierung 7-2: die Programmschleife mit Lizenzkontrolle**

## 7.3 Die Kontrollmethode (nicht für MDI_PRJ)

In der Kontrollmethode wird für die Einzelversionen die Lizenzkontrolle durchgeführt.

Wenn der Kontrollablauf korrekt abläuft, wird die öffentliche Variable TESTVERSION false und das Programm läuft in der Endversion.

Ist bei einem Kontrollablauf der Aktivierungsschlüssel nicht korrekt, wird der Anwender über eine Messagebox darüber informiert und TESTVERSION true.

Ist bei einem Passwort-Kontrollablauf das Passwort nicht korrekt wird der Programmablauf beendet.

Die Aufrufe erfolgen in Programmschleife und zur zusätzlichen Lizenzkontrolle in Programm- und KontextMenueOnClick nach einem Pausewechsel.

**Kodierung 7-3: die Kontrollmethode für die Lizenzkontrolle**

## 7.4 Die Programmschleife für MULTITHREAD

MT_Programmschleife ist die Startmethode für einen auf OnLoad erzeugten Arbeitsthread.

Wenn der Bedienerthread keine Synchronisationsanforderung stellt, wird zur Abarbeitung des Anwendungsprogramms in einer Schleife solange Basis aufgerufen bis ProgLoop false wird.

Bei einer Programmpause oder nach einer Synchronisationsanforderung wird der Thread sofort gewechselt.

Zu Beginn wird in einer Schleifenabfrage auf die Ausführung der ersten OnResize-Botschaft mit der Erzeugung der GDI+ Graphics-Objekte gewartet.

Der Arbeitsthread wird beendet, wenn auf OnClosed ProgLoop false wird und MT_Programmschleife verlassen wurde. Vorher wird für einen korrekten Programmabschluss noch einmal die Basis-Methode aufgerufen.

Eine Lizenzkontrolle darf in dieser Methode nicht durchgeführt werden, da in einem Arbeitsthread bei einem Aufruf von Dialogformen das Programm ohne Fehlermeldung abstürzt.

**Kodierung 7-4: die Programmschleife für MULTITHREAD**

## 7.5 Die Basis-Methode

Basis durchläuft die Aufrufkontrolle und das Programm, wenn keine Multimediadatei angezeigt wird und wenn keine Programmpause ist oder bei einer auszusetzenden Pause.

Der Aufruf erfolgt ständig in Programmschleife oder MT_Programmschleife.

Struktogramm

**Kodierung 7-5: die Basis-Methode**

### 7.5.1 Die Aufrufkontrolle

Die Anzahl der Aufrufe von Aufrufkontrolle werden in jeder Sekunde neu berechnet und nicht bei einer Spielpause als gefilterter Wert ohne Schwankungen, wenn diese unter 2% bleiben, in das öffentliche Datenfeld AufrufFilterAnz eingetragen.

Aufrufkontrolle wird in Basis aufgerufen.

**Kodierung 7-6: die Aufrufkontrolle**

### 7.5.2 Das virtuelle Programm des Basisprojekts

Das Programm von dieser BASISPROJEKT-Klasse zeichnet wahlfrei geometrische Figuren in die Basisbitmap und direkt zum Display.

Programm ist eine virtuelle Methode, die von der Anwendung als Schnittstelle zum Basisprojekt überschrieben werden muss.

Programm wird ständig in Basis aufgerufen, wenn keine Multimediadatei angezeigt wird und keine Programmpause ist.

**Kodierung 7-7: das virtuelle Programm des Basisprojekts**

## 7.6 Die Pause_Methode

Die Pause_Methode bearbeitet einen neuen Pausestatus und ermöglicht einen Zugriff auf lokale Daten und Methoden.

Zum Beenden einer Bildanzeige und eines Video- und Audioablaufes werden die Ziehleisten verdeckt oder zur Anpassung der Panelgröße PanelOnResize() aufgerufen.

Des Weiteren werden bei einer beendeten Multimediaanzeige vorhandene Video- und Audioobjekte entfernt, der Basiszeitgeber gestoppt, und weitere Panel aus der Statusbar entfernt (wenn zugefügt).

Durch den Aufruf der virtuellen Methode MenuCompleteHandler wird der Statusleistentext auch für die Anwendung aktualisiert.

Nur für MULTITHREAD wird die Ausführung über einen neu erzeugten Delegaten auf den Bediener-Thread umgeleitet, wenn der Aufruf nicht aus dem Bediener-Thread erfolgte und somit InvokeRequired true zurückgibt.

Der Aufruf erfolgt beim Setzen der Eigenschaft der globalen Instanzvariablen Pause.

**Kodierung 7-8: Die Pause_Methode für das Basisprojekt**

## 7.7 Die Ausnahmeprotokollierung

Ausnahmeprotokollierung erhöht das globale Datenelement catch_Ausnahmen um eins und schreibt für die ersten 25 Ausnahmen den Ausnahmetext mit Datum und Zeit in die Datei der StreamWriter-Instanz SWriter ("Titeltext" + "_Exceptions.txt" für die Einzelversionen oder MDIFormen_Exceptions.txt für das MDI-Projekt).

**Kodierung 7-9: Ausnahmeprotokollierung**

## 7.8 Die Synchronisation der Threads

ThreadSynchroStart startet die Synchronisation des Bedienerthreads (UI-Thread) mit dem Arbeitsthread. Der Aufruf von ThreadSynchroStart darf nur im Bedienerthread erfolgen.

**Kodierung 7-10: ThreadSynchroStart**

ThreadSynchroEnde beendet die Synchronisation des Bedienerthreads (UI-Thread) mit dem Arbeitsthread. ThreadSynchroEnde **muss** nach jedem Aufruf von ThreadSynchroStart aufgerufen werden.

**Kodierung 7-11: ThreadSynchroEnde**

## 7.9 Den Arbeitsthread pausieren

Arbeitsthread_pausieren pausiert den Arbeitsthread.

In der Pausezeit wird zur Vermeidung eines Deadlock eine Synchronisation des Bedienerthreads mit dem Arbeitsthread ermöglicht.

Ein Aufruf muss im Arbeitsthread erfolgen.

**Kodierung 7-12:  Arbeitsthread_pausieren**

## 7.10 Methodenaufruf über einen Delegaten

Komponenten und somit auch Dialogforms müssen im Bedienerthread erzeugt werden. Nur hierdurch können die im Botschaftsthread ablaufenden Methoden auf die darin vorhandenen Steuerelemente zugreifen!

Vor der Erzeugung von Komponenten sollte deshalb abgefragt werden, ob die Erzeugermethode im Arbeitsthread ausgeführt wird und somit auf den Bedienerthread umgeleitet werden muss. Zur Umleitung wird die Erzeugermethode durch eine der Invoke-Methoden synchron oder asynchron als Delegat ausgeführt.

Delegaten werden in allen Projekten am Anfang der Methoden zur Erzeugung der Menüs erzeugt und ausgeführt. Des Weiteren im Basisprojekt in Animated_Bitmap zur Animation von geladenen Bitmaps, in der Pause_Methode und in Titeltext mit string-Parameter.

**Kodierung 7-13: Methodenaufruf über einen Delegaten**

# 8 Die multilingualen Basismenüs

Das Hauptmenü für die Form des Basisprogramms wird multilingual in ErzeugeFormMenues zusammengestellt. Zum Abschluss der Methode wird das Kontextmenü durch Aufruf von ErzeugeKontextMenue entworfen.

Die Anwendung kann durch Überschreiben der virtuellen Methoden mit vorangehendem Aufruf der Basismethode die Elemente beider Menüs verändern und erweitern.

Die einzelnen Menüelemente werden über die Einträge in den dreidimensionalen Menütext-Arrays zusammengestellt. Hierbei bestimmt die Größe der zweiten Array-Dimension die Anzahl der Menüelemente. Die erste Dimension wählt die Textsprache und die dritte Dimension den Text selbst aus.

Die Menüs werden nach dem Programmstart auf OnLoad erstmalig zur Verfügung gestellt. Im weiteren Programmablauf müssen die Menüs nach einem Sprachwechsel in SpracheMenueOnClick und in DialogResultOK, nachdem der Basis-Dialog geschlossen wurde, neu erzeugt werden.

Nach einem Wechsel der Hilfetextsprache in HilfeTextSpracheMenueOnClick und einer gewechselten Statusbaranzeige in BasisMenueOnClick korrigiert ein neuer Entwurf nur die Checkmarke. Weitergehend kann hierdurch aber das gesamte Menüelement durch die Anwendung verschoben werden.

Zur Aktualisierung der Titelleiste der Form wird Titeltext aufgerufen, was nach einem Wechsel der Bedienersprache erforderlich ist. Danach wird der Panel- und Tooltiptext der Statusbar multilingual korrigiert.

Für alle Menüelemente werden Handler für die Select-Ereignisse und, wenn erforderlich, für die Click-Ereignisse eingetragen.

**Hinweis 9 zur Menübeschreibung!**

Im weiteren Ablauf werden die Begriffe Menüelement und Menüpunkt gleichbedeutend verwendet. Ein Menüelement kann selbst wieder aus mehreren Elementen bestehen. Der Index gibt den Menüpunkt aus mehreren zusammengehörenden Elementen von null beginnend an.

## 8.1 Der Basisentwurf des Hauptmenüs

Das Hauptmenü wird nur der Form zugewiesen, wenn das öffentliche Datenfeld BasisHauptMenueAnzeige true ist.

Dem Menüpunkt Pause wird die Taste F3 zur schnellen Aktivierung zugewiesen.
Nicht für das MDI-Projekt wird mit F1 die Einführung des Hilfetextes angezeigt.

Weitere Shortcuts sind den Menüpunkten zur Anzeige der Dialogforms zugewiesen.

nur für MULTITHREAD gilt, wenn ErzeugeFormMenues nicht in ihrem Erzeuger-Thread (UI-Thread) aufgerufen wird, wird sie durch ihre Delegat-Methode im Erzeuger-Thread ausgeführt.

**Kodierung 8-1: ErzeugeFormMenues für das Basisprogramm**

## 8.2 Der Basisentwurf des Kontextmenüs

ErzeugeKontextMenue entwirft das multilinguale Kontextmenü für die Form des Basisprogramms.

Die Anwendung kann durch Überschreiben der virtuellen Methode mit vorangehendem Aufruf der Basismethode das Kontextmenü verändern und erweitern.

Der Aufruf erfolgt in ErzeugeFormMenues nach dem Entwurf des Basis-Hauptmenüs.

**Kodierung 8-2: ErzeugeKontextMenue für das Basisprogramm**

## 8.3 Bearbeitung der Basismenü-Ereignisse

### 8.3.1 Das Startereignis für das Basismenü

MenuStartHandler hat vorerst keine Aufgabe.

Der Aufruf erfolgt als eingetragener Handler für das MenuStart-Ereignis der Form, wenn die Menüauswahl startet.

**Kodierung 8-3: MenuStartHandler für das Basisprogramm**

### 8.3.2 Das Abschlussereignis für das Basismenü

In MenuCompleteHandler wird für eine angezeigte Multimediadatei der Dateiname angezeigt. Sonst wird der Starttext des Basisprogramms zum ersten Panel der StatusBar ausgegeben.

Der Aufruf erfolgt als eingetragener Handler für das MenuComplete-Ereignis der Form, nachdem die Menüauswahl abgeschlossen wurde.

Für das MDI-Menü ruft das MDI-Verwaltungsprogramm MenuCompleteHandler auf OnMenuComplete für das aktive Kindfenster explizit ein weiteres Mal auf, weil das aktive Kindfenster nicht benachrichtigt wird.

**Kodierung 8-4: MenuCompleteHandler für das Basisprogramm**

### 8.3.3 Hauptmenüelement-Ereignisse für das Basisprogramm

HauptMenueOnSelect gibt den Hilfetext für den ausgewählten Hauptmenüpunkt zum ersten Panel der StatusBar aus.

Der Aufruf erfolgt als ein in ErzeugeFormMenues für Select-Ereignisse eingetragener Handler, wenn ein Hauptmenüelement ausgewählt wurde.

**Kodierung 8-5: HauptMenueOnSelect für das Basisprogramm**

### 8.3.4 Programmmenüelement-Ereignisse für das Basisprogramm

ProgrammMenueOnClick bearbeitet die Auswahl der Programm-Menüelemente, die von dem Basisprojekt erzeugt wurden. Der public-Zugriffsmodifizierer ist wegen direkter Methodenaufrufe erforderlich.

Bei einem Pausewechsel wird für die Endversion in der Kontrollmethode nochmals eine Lizenzkontrolle durchgeführt (nicht für MDI_PRJ).

Der Aufruf erfolgt als ein in ErzeugeFormMenues für Click-Ereignisse eingetragener Handler, wenn ein Programm-Menüelement angeklickt wurde.

**Kodierung 8-6: ProgrammMenueOnClick für das Basisprogramm**

ProgrammMenueOnPopup ist der in ErzeugeFormMenues eingetragene Handler zum Ausschalten von Programm-Menüpunkten, der inhaltslos zum Überschreiben für die Anwendung definiert wird.

Der Aufruf erfolgt als ein in ErzeugeFormMenues für Popup-Ereignisse eingetragener Handler, bevor der Programm-Menüpunkt ausgeklappt wird.

**Kodierung 8-7: ProgramMenueOnPopup für das Basisprogramm**

ProgrammMenueOnSelect gibt den Hilfetext für den ausgewählten Programm-Menüpunkt zum ersten Panel der StatusBar aus. Der Aufruf erfolgt als ein in ErzeugeFormMenues für Select-Ereignisse eingetragener Handler, wenn ein Programm-Menüelement ausgewählt wurde.

**Kodierung 8-8: ProgrammMenueOnSelect für das Basisprogramm**

### 8.3.5 Kontextmenüelement-Ereignisse für das Basisprogramm

KontextMenueOnClick bearbeitet die Clicks der Kontext-Menüelemente, die von dem Basisprojekt erzeugt wurden.

Bei einem Pausewechsel wird für die Endversion in der Kontrollmethode nochmals eine Lizenzkontrolle durchgeführt (nicht für MDI_PRJ).

Der Aufruf erfolgt als ein in ErzeugeFormMenues für Click-Ereignisse eingetragener Handler, wenn ein Kontext-Menüelement angeklickt wurde.

**Kodierung 8-9: KontextMenueOnClick für das Basisprogramm**

KontextMenueOnPopup ist der in ErzeugeFormMenues eingetragene Handler zum Deaktivieren von Kontext-Menüpunkten, der inhaltslos zum Überschreiben für die Anwendung definiert wird.

Der Aufruf erfolgt als ein in ErzeugeFormMenue für Popup-Ereignisse eingetragener Handler, bevor das Kontextmenü angezeigt wird.

**Kodierung 8-10: KontextMenueOnPopup für das Basisprogramm**

KontextMenueOnSelect gibt den Hilfetext für den ausgewählten Kontext-Menüpunkt zum ersten Panel der StatusBar aus. Der Aufruf erfolgt als ein in ErzeugeFormMenues für Select-Ereignisse eingetragener Handler, wenn ein Kontext-Menüelement ausgewählt wurde.

**Kodierung 8-11: KontextMenueOnSelect für das Basisprogramm**

### 8.3.6 Basismenüelement-Ereignisse für das Basisprogramm

BasisMenueOnClick bearbeitet die Clicks der Basis-Menüelemente, die von dem Basisprojekt erzeugt wurden.

Der Aufruf erfolgt als ein in ErzeugeFormMenues für Click-Ereignisse eingetragener Handler, wenn ein Basis-Menüelement angeklickt wurde.

Nach Auswahl von "Basis" wird die Dialogform des Basisprojekts modal erzeugt, angezeigt und wieder entfernt.

Nach Auswahl von "Drucken" wird in BasisBitmapDrucken die Basisbitmap ausgedruckt. Nach Auswahl von "Speichern" wird in BasisBitmapSpeichern die Basisbitmap in eine Grafikdatei abgespeichert.

Nach Auswahl von "Laden" wird die Basisbitmap aus einer Grafikdatei erzeugt oder eine Audio- oder Videodatei abgespielt.

Nach Auswahl von Originalgröße wird das Bild  oder das Video in der Originalgröße angezeigt bzw. abgespielt.

"Statusleiste" zeigt oder verdeckt die Statusleiste mit gewechselter StBarAnzeige.

**Kodierung 8-12: BasisMenueOnClick für das Basisprogramm**

BasisMenueOnPopup ist der in ErzeugeFormMenues eingetragene Handler zum Ausschalten von Basis-Menüpunkten.

Der Aufruf erfolgt als ein in ErzeugeFormMenues für Popup-Ereignisse eingetragener Handler, bevor der Basis-Menüpunkt ausgeklappt wird.

**Kodierung 8-13: BasisMenueOnPopup für das Basisprogramm**

BasisMenueOnSelect gibt den Hilfetext für den ausgewählten Basis-Menüpunkt zum ersten Panel der StatusBar aus. Der Aufruf erfolgt als ein in ErzeugeFormMenues für Select-Ereignisse eingetragener Handler, wenn ein Element des Basis-Menüs ausgewählt wurde.

**Kodierung 8-14: BasisMenueOnSelect für das Basisprogramm**

## 8.3.7 Hilfemenüelement-Ereignisse für das Basisprogramm

HilfeMenueOnClick bearbeitet die Clicks der Hilfe-Menüelemente, die von dem Basisprojekt erzeugt wurden.

Der Aufruf erfolgt als ein in ErzeugeFormMenues für Click-Ereignisse eingetragener Handler, wenn ein Hilfe-Menüelement angeklickt wurde.

Nach Auswahl von "Einführung" und "Menübefehle" wird aus den multilingualen HTML-Hilfetextdateien ein Inhaltsverzeichnis mit dem Einführungstext oder der Beschreibung der Menübefehle aufgerufen.

Nach Auswahl von "über..." wird eine Dialogform mit multilingualer Information über das Basisprogramm angezeigt.

**Kodierung 8-15: HilfeMenueOnClick für das Basisprogramm**

HilfeMenueOnSelect gibt den Hilfetext für den ausgewählten Hilfe-Menüpunkt zum ersten Panel der StatusBar aus. Der Aufruf erfolgt als ein in ErzeugeFormMenues für Select-Ereignisse eingetragener Handler, wenn ein Hilfe-Menüelement ausgewählt wurde.

**Kodierung 8-16: HilfeMenueOnSelect für das Basisprogramm**

### 8.3.8 Sprachemenüelement-Ereignisse für das Basisprogramm

SpracheMenueOnClick bearbeitet die Clicks der Sprache-Menüelemente, die von dem Basisprojekt erzeugt wurden.

Der Aufruf erfolgt als ein in ErzeugeFormMenues für Click-Ereignisse eingetragener Handler, wenn ein Sprache-Menüelement angeklickt wurde.

Nach Anwahl eines Sprache-Menüpunktes werden die öffentlichen Datenfelder User_Language und Help_Language verändert und das Menü durch Aufruf von ErzeugeFormMenues in der neuen Sprache erzeugt. Der Paneltext der StatusBar und der Formtitel werden hierbei in der neuen Sprache aktualisiert.

**Kodierung 8-17: SpracheMenueOnClick für das Basisprogramm**

SpracheMenueOnSelect gibt den Hilfetext für den ausgewählten Sprache-Menüpunkt zum ersten Panel der StatusBar aus. Der Aufruf erfolgt als ein in ErzeugeFormMenues für Select-Ereignisse eingetragener Handler, wenn ein Sprache-Menüelement ausgewählt wurde.

**Kodierung 8-18: SpracheMenueOnSelect für das Basisprogramm**

### 8.3.9 Hilfetextsprachemenüelement-Ereignisse für das Basisprogramm

HilfeTextSpracheMenueOnClick bearbeitet die Clicks der Hilfetextsprache-Menüelemente, die von dem Basisprojekt erzeugt wurden.

Der Aufruf erfolgt als ein in ErzeugeFormMenues für Click-Ereignisse eingetragener Handler, wenn ein Hilfetextsprache-Menüelement angeklickt wurde.

Nach Anwahl eines Hilfetextsprache-Menüpunktes wird das öffentliche Datum Help_Language verändert und das Menü durch Aufruf von ErzeugeFormMenues neu erzeugt. Hierdurch kann das ganze Menüelement durch die Anwendung verschoben werden.

**Kodierung 8-19: HilfeTextSpracheMenueOnClick für das Basisprogramm**

HilfeTextSpracheMenueOnSelect gibt den Hilfetext für den ausgewählten Hilfetextsprache-Menüpunkt zum ersten Panel der StatusBar aus. Der Aufruf erfolgt als ein in ErzeugeFormMenues für Select-Ereignisse eingetragener Handler, wenn ein Hilfetextsprache-Menüelement ausgewählt wurde.

**Kodierung 8-20: HilfeTextSpracheMenueOnSelect für das Basisprogramm**

# 9 Speichern und Drucken der BasisBitmap

## 9.1 Abspeichern der Basisbitmap in eine Datei

BasisBitmapSpeichern speichert die Basisbitmap in eine Datei nach Auswahl des Dateinamens über die Standard-Dialogform "Speichern unter".

Der Aufruf erfolgt in BasisMenueOnClick und KontextMenueOnClick nach der Menüauswahl "Speichern...".

Die Dateierweiterungen .wmf, .emf, .exif, .mbmp und .ico werden nicht als Dateityp in der Speichern-Dialogform aufgeführt, weil bei einer Auswahl dieser Dateitypen die Datei immer im Standard .png-Format abgespeichert wird.

**Kodierung 9-1: BasisBitmapSpeichern**

## 9.2 Ausgabe der Basisbitmap zum Drucker

BasisBitmapDrucken startet eine Druckerausgabe nach Auswahl eines Druckers über die Standard-Dialogform "Drucken" oder, wenn nicht erfolgreich, über die Seitenvorgabe- und Druckervorschau-Dialoge.

BasisBitmapDrucken wird erst nach der gesamten Datenübertragung zum Drucker einschließlich einer nachfolgenden Druckervorschau verlassen.

Ein Aufruf erfolgt in BasisMenueOnClick und KontextMenueOnClick nach Menüauswahl "Drucken...".

**Hinweis 10 zur Vorschau der Druckerausgabe**

Eine Vorschau der Druckerausgabe wird nach Anzeige der Standard-Dialogform "Drucken" durch den Druckertreiber selbst durchgeführt. Die Standard-Dialogform "Drucken" wird unter den 64-Bit Windows-Versionen nur als extended-Dialog angezeigt.

**Kodierung 9-2: BasisBitmapDrucken**

### 9.2.1 Ausgabe der Druckerseiten

OnPrintPage gibt die einzelnen Seiten der Basisbitmap zum ausgewählten Drucker aus.

Der Aufruf erfolgt als ein in BasisBitmapDrucken eingetragener Handler für das PrintPage-Ereignis des Drucker-Dokuments.

**Kodierung 9-3: OnPrintPage**

# 10 Laden von Image-, Audio- und Videodateien

In Image_Audio_Video_Laden wird nach Auswahl des Dateinamens über die Standard-Dialogform "Öffnen" eine Bilddatei in die Basisbitmap geladen.

Von dem ausgewählten Bild wird eine Bitmap mit dem zugehörigen Grafik-Objekt erzeugt. Hierbei wird auch die Wiedergabe animierter Bitmaps unterstützt.

Eine ausgewählte Videodatei wird in der Panelform abgespielt. Eine Audiodatei ist nur akustisch hörbar.

Zur Aktualisierung der horizontalen Ziehleiste und der Panellleiste wird der Basis-Zeitgeber gestartet.

Die Aufrufe erfolgen in BasisMenueOnClick nach der Menüauswahl "Laden..." und nach dem Programmstart auf OnLoad zur sofortigen Anzeige einer Multimediadatei mit dem Parameter Dialoganzeige false.

Durch einen Aufruf in PanelOnDragDrop mit Dialoganzeige false, wird die zuvor in DialogFileName eingetragene Datei angezeigt. Hierdurch wird Drag and Drop realisiert.

Bei einer Programmausnahme wird eine Messagebox angezeigt. Danach werden durch Aufruf von PanelOnResize normale Grafikobjekte erzeugt.

**Kodierung 10-1: Image_Audio_Video_Laden**

## 10.1.1 Wiedergabe einer animierten Bitmap

Animated_Bitmap gibt den nächsten Rahmen einer geladenen animierten Basisbitmap zur Form aus.

Der Aufruf erfolgt als Event-Handler für die animierte Bitmap aus Image_Audio_Video_Laden.

Die Ausführung wird über einen neu erzeugten Delegaten auf den Bediener-Thread umgeleitet, wenn InvokeRequired true zurückgibt.

**Kodierung 10-2: Animated_Bitmap**

## 10.1.2 Der Basiszeitgeber

In Basis_Zeitgeber wird bei einer ablaufenden Video- oder Audiodatei die horizontale Ziehleiste aktualisiert und Zeit- oder Bildinformationen in das zweite Panel der Statusleiste eingetragen. Für den Videoablauf wird noch das dritte Panel mit der Videogröße beschrieben.

Der Aufruf erfolgt als Event-Handler eines in OnLoad erzeugten und in Image_Audio_Video_Laden gestarteten Sekunden-Zeitgebers.

**Kodierung 10-3: Basis_Zeitgeber**

## 10.1.3 Überprüfen der Multimediadatei

In Multimediadatei_OK wird geprüft, ob die in MMDateiName abgelegte Datei angezeigt oder abgespielt werden kann.

Der Aufruf erfolgt in PanelOnDragEnter.

**Kodierung 10-4: Multimediadatei_OK**

## 10.1.4 Ziehen und Ablegen von Dateien mit der Maus

Zur Realisierung von Drag and Drop, werden in BASISPROJEKT.ErzeugeFormPanel zwei Handler für das DragEnter- und das DragDrop-Ereignis des Panels eingerichtet. Hierdurch werden die Ereignismethoden PanelOnDragEnter und PanelOnDragDrop aufgerufen.

In PanelOnDragEnter wird durch Multimediadatei_OK überprüft, ob es sich um eine anzeigbare Datei handelt. Wenn dies der Fall ist, wird der DragDrop-Effektcursor auf Copy gesetzt. Nur dann wird beim Ablegen der Datei eine Botschaft nach PanelOnDragDrop gesendet.

In PanelOnDragDrop wird durch den Aufruf von Image_Audio_Video_Laden mit dem Parameter Dialoganzeige false, die zuvor in DialogFileName eingetragene Datei angezeigt.

# 11 Verwaltung der Ziehleisten

Zur Verwaltung der Ziehleisten für eine angezeigte Multimediadatei werden in der BASIS_PROCS-Klasse die Methoden Rollbalkenverw, VertRollbalken und HorzRollbalken zur Verfügung gestellt.

## 11.1 Die Ziehleisten einstellen

In Rollbalkenverw werden die Positionsgrenzen und Abgriffpositionen der horizontalen und vertikalen Ziehleisten zur aktuell in der Form angezeigten Bitmap oder zur abgespielten Video- oder Audiodatei des Basisprojekts eingestellt.

Ein Aufruf erfolgt für eine angezeigte Multimediadatei auf PanelOnResize, nach einem Wechsel von und zur Originalgröße und nach einer neu geladenen Datei in Image_Audio_Video_Laden.

nur für MULTITHREAD gilt, wenn Rollbalkenverw nicht in ihrem Erzeuger-Thread (Bedienerthread) aufgerufen wird, wird sie durch ihre Delegat-Methode im Erzeuger-Thread ausgeführt.

**Kodierung 11-1: Rollbalkenverw für die Basisbitmap**

## 11.2 Die Position der vertikalen Ziehleiste aktualisieren

Die Methode VertRollbalken aktualisiert nach einer Bewegung der vertikalen Ziehleiste unter Beachtung der Ober- und Untergrenze die Positionsangaben.

Zur Aktualisierung der Grafikanzeige wird ein PanelOnPaint-Ereignis aktiviert.

Der Aufruf erfolgt auf PanelOnScroll für den vertikalen Rollbalken.

**Kodierung 11-2: VertRollbalken für das Basisprogramm**

## 11.3 Die Position der horizontalen Ziehleiste aktualisieren

Die Methode HorzRollbalken aktualisiert nach einer Bewegung der horizontalen Ziehleiste unter Beachtung der Ober- und Untergrenze die Positionsangaben.

Zur Aktualisierung der Grafikanzeige wird ein PanelOnPaint-Ereignis aktiviert.

Der Aufruf erfolgt auf PanelOnScroll für den horizontalen Rollbalken.

**Kodierung 11-3: HorzRollbalken für das Basisprogramm**

# 12 Die Dialogform des Basisprogramms

Die BASIS_DIALOG-Klasse realisiert eine dreiseitige modale Dialogform mit TabControls zur Seitenauswahl durch Vererbung von der Form-Klasse erstellt wird. Mit dem Basisdialog können grundlegende Einstellungen für das aktive Programm vorgenommen werden.

Auf der Basisseite kann das Hauptmenü und die Statusleiste ein- und ausgeblendet werden, die Sichbarkeit der Form eingestellt und ein transparenter Hintergrund für die Grafikfläche der Form ausgewählt werden. Zum Programmende kann die Dialoganzeige mit den Ausnahmen abgewählt werden.

Auf der Grafikseite kann die Qualität der Grafikausgabe und der Pixelausgleich bei der Darstellung ausgewählt werden.

Innerhalb der Spracheseite kann die Sprachauswahl für die Bediener- und die Hilfetextsprache vorgenommen werden.

**Kodierung 12-1: Die BASIS_DIALOG-Klasse**

```
///*************************************************************************
/// Version 4.0 vom 11.04.18
///*************************************************************************
[CLSCompliantAttribute(true)]    // markiert die Klasse als CLS-kompatibel
public class BASIS_DIALOG : Form
{

    #region Datenfelder und Konstruktor #endregion

    #region InitializeComponent #endregion

    #region Bearbeitung der Standard-Ereignisse für den Dialog #endregion

    #region Ereignis-Handler für die Kontrollen #endregion

}
```

## 12.1 Die Instanzvariablen der Basisdialogform

Die Dialogkontrollen wurden von dem Designer bei der Erstellung der Dialogformen zugefügt und dürfen nicht manuell geändert werden.

**Kodierung 12-2: Die Instanzvariablen der BASIS_DIALOG-Klasse**

## 12.2 Der Konstruktor der Basisdialogform

In InitializeComponent werden die mit dem Designer erstellten Basis-Dialogseiten aufgebaut und eine Instanz von BASIS_DIALOG_TEXTE erstellt. Der Aufruf erfolgt als Konstruktor für die BASIS_DIALOG-Klasse nach Menüpunktauswahl "Basis..." in BasisMenueOnClick und KontextMenueOnClick.

**Kodierung 12-3: Der Konstruktor der Basisdialogform**

## 12.3 Die Basisdialogform mit dem Designer erzeugen

InitializeComponent erzeugt die Basis-Dialogseiten mit den Kontrollen. Der Aufruf erfolgt im BASIS_DIALOG-Konstruktor.

Die Kodierung wird automatisch mit dem Designer in deutscher Sprache erzeugt und darf nicht manuell geändert werden.

BASIS_DIALOG_HelpRequested ist als Handler für das HelpRequested-Ereignis der Dialogform eingetragen und Info_Button_Click ist der Handler für das Click-Ereignis des Info-Button.

```
///*************************************************************************
///  Version 4.0 vom 09.04.18
///*************************************************************************
private void InitializeComponent()
{...}
```

### 12.3.1 Veränderungen am Design

Zur besseren Bedienung der Dialogform auf hochauflösenden Displays wurde für die Version 4.0 die Schrift für alle Dialogseiten auf Microsoft Sans Serif 10 (9,75 Pt) vergrößert (vorher 8,25Pt). Hierzu wurde die spanische Beschreibung für die Abbrechen-Taste auf Cancellar verändert (Franz. Annuler).

Auf der Basis-Seite wurden die Auswahlfelder etwas nach unten verschoben und die Groupbox verbreitert. Die Textbox für die Sichtbarkeit wurde an die Position 12x22 verschoben und für die neue Fontgröße 11 (11,25 Pt) auf 77x24 vergrößert. Die Über-Taste wurde leicht vergrößert.

Auf der Grafik-Seite wurden beide Groupboxen vergrößert und verschoben und die Optionsfelder nach unten verschoben. Hierdurch kann der Text der Groupbox auf zwei Zeilen angezeigt werden.

Auf der Sprache-Seite wurden beide Groupboxen erhöht und die Optionsfelder nach unten verschoben. Der deutsche Text „Französisch" zu „Französ." und für Englisch und Italienisch „Portugiesisch" zu „Portug." abgekürzt.

**Hinweis 11 zu den Design-Veränderungen der Basis-Dialogform**

auf einem Tablet-PC mit einer Displaygröße von 1280x800 wird der Basisdialog derart verkleinert dargestellt, dass die Texte zum Teil nicht mehr vollständig angezeigt werden.

## 12.4 Mehrsprachige Basisdialogtexte

Die Dialogtexte werden mehrsprachig in der Klasse BASIS_DIALOG_TEXTE zur Verfügung gestellt. Sie bestehen aus dem Titeltext, den Texten für die TabControls und Messageboxtexten, die alle ohne Tooltips auskommen.

Die weiteren Texte für die Kontrollen aller Dialogseiten enthalten Tooltips und wenn erforderlich zusätzlich erweiterte Hilfetexte.

Eine Instanz von BASIS_DIALOG_TEXTE wird im Konstruktor der BASIS_DIALOG-Klasse erzeugt.

**Hinweis 12 zu den Tooltipps!**

Wenn ein erweiterter Hilfetext zur Verfügung steht, wird dem Tooltip ein Hinweis in der Form (-> ?-Hilfe) zugefügt.

Zur Seiteneinsparung werden nachfolgend nur die deutschen und englischen Texte aufgeführt.

**Kodierung 12-4: Die BASIS_DIALOG_TEXTE-Klasse**

## 12.5 Bearbeitung der Ereignisse für die Basisdialogform

### 12.5.1 OnLoad für die Basisdialogform

OnLoad initialisiert die Kontrollen mit Daten und multilingualen Texten. Des Weiteren werden erweiterte Hilfetexte und Tooltipps eingeführt.

OnLoad überschreibt die geerbte Form-Methode und wird nur einmal aufgerufen, bevor die Dialogform angezeigt wird.

**Kodierung 12-5: OnLoad für die Basisdialogform**

### 12.5.2 BASIS_DIALOG_HelpRequested

BASIS_DIALOG_HelpRequested zeigt eine Messagebox mit dem ausführlichen Hilfetext der ausgewählten Kontrolle an. Der Aufruf erfolgt als ein im Designer eingetragener Handler für das HelpRequested-Ereignis der Basis-Dialogform nach Anwahl einer Kontrolle über den ?-Button oder die F1-Taste.

Das Ereignis wird auch ausgelöst, wenn der Kontrolle über die HelpProvider-Klasse ein Popup-Hilfetext zugewiesen wurde und dieser durch SetShowHelp für die Kontrolle deaktiviert ist.

Achtung! Das Ereignis wird nicht mehr ausgelöst, nachdem einmal ein Popup-Hilfetext angezeigt wurde.

**Kodierung 12-6: BASIS_DIALOG_HelpRequested**

### 12.5.3 OnClosed für die Basisdialogform

OnClosed ruft zur Realisierung der Dialog-Einstellungen DialogResultOK auf.

Für MULTITHREAD wird vor der Veränderung der globalen Daten in DialogResultatOK der Bedienerthread mit dem Arbeitsthread synchronisiert.

OnClosed überschreibt die geerbte Form-Methode und wird nur einmal aufgerufen, nachdem die Dialogform geschlossen wurde.

**Kodierung 12-7: OnClosed für die Basisdialogform**

## 12.6 Schließen der Dialogform mit dem OK-Button

DialogResultOK realisiert alle Dialog-Einstellungen auf Schließen des Dialoges mit dem OK-Button und erzeugt in ErzeugeFormMenues neue Menüs.

Der Aufruf erfolgt in OnClosed bevor die modale Dialog-Form geschlossen wird.

**Kodierung 12-8: DialogResultOK für die Basisdialogform**

## 12.7 Ereignisse für die Kontrollen der Basisdialogform

### 12.7.1 InfoButton_Click für die Basisdialogform

InfoButton_Click zeigt die Ueber-Dialogform mit Informationen über das Basisprojekt und das Betriebssystem an.

Der Aufruf erfolgt als ein eingetragener Handler für das Click-Ereignis des "über"...-Button von der Basis-Dialogseite.

**Kodierung 12-9: InfoButton_Click für die Basisdialogform**

### 12.7.2 HilfeButton_Click für die Basisdialogform

In HilfeButton_Click wird für die aktuelle Dialogseite die multilinguale HTML-Hilfe aufgerufen. Der Aufruf erfolgt als eingetragener Handler für das Click-Ereignis des Hilfe-Button der Basis-Dialogform.

**Hinweis 13 zu den Hilfetexten für die Basisdialogseiten!**

Die Hilfe für die Basisdialogseiten wird über Tooltips und erweiterte Hilfetexte realisiert. Zur Zeit ist kein Hilfe-Button in die Dialogform eingebunden und es sind auch keine Hilfetexte für die Dialogseiten in den HTML-Hilfedateien vorhanden.

**Kodierung 12-10: HilfeButton_Click für die Basisdialogform**

# 13 Standard Dialogformen des Basisprojekts

Zur Anzeige von Programminformationen stellt das Basisprojekt die folgenden, mit dem Designer erstellten und zu verwaltenden Dialogformen zur Verfügung.

## 13.1 Das Infofeldformular

Diese Dialogform wurde durch Auswahl von Windows Form / Infofeld zum Basisworkspace als neues Element hinzugefügt. Dadurch werden auch die Quellcodedateien AboutBox.cs, AboutBox.Designer.cs und AboutBox.resx automatisch zum Projekt hinzugefügt.

In dem Infofeldformular werden die in der AssemblyInfo-Datei eingetragenen Projekt-Informationen aufbereitet angezeigt. Dies sind im Einzelnen von der Datei AssemblyInfo.cs die Attribute AssemblyTitle, AssemblyProduct, AssemblyVersion, AssemblyCopyright, AssemblyCompany und AssemblyDescription.

Das Infofeldformular wird von der Anwendung 1 und dem Formen-Verwaltungsprogramm nach Menüauswahl „Hilfe / über" zur Anzeige von Programminformationen aufgerufen.

## 13.2 Die Über-Dialogform

Als Alternative zu dem Infofeldformular steht eine Dialogform mit einer Ausgabetextbox und den Button „OK" und „WWW" zur Verfügung. Die Anzeige wird in den Quellcodedateien Ueber.cs, Ueber.designer.cs und Ueber.resx realisiert.

In das Textfeld werden der BasisText1 mit Informationen über die Version des Basisprojekts und Copyrights ausgegeben. Des Weiteren der vollständige Assemblyname des Programms, die verwendete Common Language Runtime CLR, die Betriebssystemversion und die Bitanzahl der .Net Platform, die über die Länge eines int-Pointers ermittelt wird.

Über den „WWW"-Button wird die Entwicklerseite von jk-ware aufgerufen. Hierzu wird auch eine erforderliche Einwahl in das Internet durchgeführt.

Diese Dialogform wird von dem Basisprogramm nach Menüauswahl von „Hilfe / über..." und als „über..."-Button auf der Basis-Dialogseite zur Anzeige von Informationen über das Basisprojekt verwendet.

## 13.3 Ein Informationsdialog

Als Alternative zu der Messagebox des .Net-Frameworks wird eine einfache Dialogform mit einer Ausgabetextbox und einem OK-Button zur Verfügung gestellt. Die Anzeige wird in den Dateien Info.cs, Info.Designer.cs und Info.resx realisiert.

Der Informationsdialog wird innerhalb des Kontrollablaufes für den Lizenzschutz bei einem fehlenden Aktivierungsschlüssel zur Informationsanzeige verwendet.

# 14 Der Lizenzschutz für die Programme

In der LIZENZSCHUTZ-Klasse wird nach einer Passworteingabe in einer binären Kontrolldatei ein Programmpfad verglichen oder in einer zugesandten Lizenzschutzdatei ein kryptographischer Schlüssel kontrolliert.

Der Lizenzschutz wird innerhalb der Quellcodedatei Basis_Procs.cs realisiert.

Für die Version 4.0 werden im Kontrollablauf die BReader- und BWriter-Instanzen, und dadurch auch der Filestream FS, in einem finally-Block freigegeben. Des Weiteren wurden in den Sprachauswahl- und Lizenzvertrag-Dialogformen einige Schalter zur korrekten Textanzeige vergrößert.

**Kodierung 14-1: LIZENZSCHUTZ-Klasse**

```
#region Methoden und Dialogformen für den Lizenzschutz
///*************************************************************************
/// Version 4.0 vom 01.04.18
///*************************************************************************
[CLSCompliantAttribute(true)]    // markiert die Klasse als CLS-kompatibel
public class LIZENZSCHUTZ
{

    #region Datenfelder und Konstruktor #endregion

    #region Methoden für den Kontrollablauf #endregion

    #region Dialogformen für den Lizenzschutz #endregion

}
#endregion
```

## 14.1 Instanzvariable und Konstruktor

**Kodierung 14-2: Die Instanzvariablen der LIZENZSCHUTZ-Klasse**

**Kodierung 14-3: LIZENZSCHUTZ-Konstruktor**

## 14.2 Der Kontrollablauf

Der Kontrollablauf kontrolliert bei einer Passwortabfrage in der Kontrolldatei einen abgelegten Programmpfad oder bei einer Lizenzabfrage in der Lizenzdatei einen Aktivierungsschlüssel.

Ist das Programmpasswort nicht korrekt, werden Dialogformen zur Sprachauswahl, zur Akzeptanz eines Lizenzvertrages („EULA“) und zur Eingabe des Passwortes ausgegeben.

Bei einer Lizenzabfrage wird bei einer fehlenden Lizenzdatei oder einem unkorrekten Aktivierungsschlüssel innerhalb der Lizenzdatei, nach dem Dialog zur Akzeptanz des Lizenzvertrages („EULA“), ein Dialog mit dem Registrierungstext angezeigt.

**Kodierung 14-4: Der Kontrollablauf**

## 14.3 Die Lizenzdateikontrolle

In Lizenzdateikontrolle wird der Aktivierungsschlüssel in der Lizenzdatei unter Filename überprüft.

Wenn die Lizenzdatei nicht vorhanden ist oder der Aktivierungsschlüssel unkorrekt ist, werden nach dem Dialog zur Sprachauswahl noch Dialoge zur Akzeptanz oder Ablehnung eines Lizenzvertrags („EULA") und zum Kopieren des Registrierungsschlüssels angezeigt.

Der Aufruf erfolgt in Kontrollablauf.

**Kodierung 14-5: Lizenzdateikontrolle**

## 14.4 Regenerieren des Aktivierungsschlüssels

Dekrypt_Aktivierungsschluessel regeneriert den Aktivierungsschlüssel aus den Daten der Lizenzdatei in Dateiname.

Der Aufruf erfolgt in Lizenzdateikontrolle.

**Kodierung 14-6: Dekrypt_Aktivierungsschluessel**

## 14.5 Dialogformen für den Lizenzschutz

Für einen Zugriff auf die gemeinsamen Instanzvariablen für den Lizenzschutz, sind alle Klassen für die Dialogformen innerhalb der LIZENZSCHUTZ-Klasse aufgeführt.

Die Dialogformen für den Lizenzschutz wurden alle manuell ohne Designer-Unterstützung entworfen.

Die Dialogform zur Eingabe eines Aktivierungsschlüssels wird innerhalb des Lizenzschutzablaufs nicht eingesetzt, weil der Aktivierungsschlüssel in einer Lizenzdatei enthalten ist, die an den Kunden übermittelt wird.

Nachfolgend werden die Quelltexte hauptsächlich sich selbst erläuternd aufgelistet.

Zur Einsparung von Seiten wird die Initialisierung der String-Arrays nur in den Sprachen Deutsch und Englisch aufgeführt.

### 14.5.1 Die Dialogform zur Sprachauswahl

Die SPRACHE_DIALOG-Klasse erzeugt eine Dialogform zur Sprachauswahl.

**Kodierung 14-7: Die SPRACHE_DIALOG-Klasse**

```
///*************************************************************************
/// Version 4.0 vom 01.04.18
///*************************************************************************
[CLSCompliantAttribute(true)]   // markiert die Klasse als CLS-kompatibel
protected class SPRACHE_DIALOG : Form
{

    #region Datenfelder, Konstruktor und InitializeComponent #endregion
    #region Bearbeitung der Ereignisse für die Sprachauswahl-Dialogform #endregion

}
```

#### 14.5.1.1 Die Instanzvariablen

**Kodierung 14-8: Die Instanzvariablen**

#### 14.5.1.2 Der Konstruktor

Der Konstruktor für die SPRACHE_DIALOG-Klasse initialisiert ein zweidimensionales Array mit multilingualen Texten für die Kontrollen und erstellt die Sprache-Dialogform in InitializeComponent.

**Kodierung 14-9: Der Konstruktor**

#### 14.5.1.3 InitializeComponent

In InitializeComponent wird die Sprache-Dialogform mit den Kontrollen aufgebaut. Die Codierung wurde manuell in deutscher Sprache durchgeführt.

Der Aufruf erfolgt im Konstruktor.

**Kodierung 14-10: InitializeComponent**

#### 14.5.1.4 Bearbeitung der Ereignisse für die Sprachedialogform

OnLoad initialisiert die Kontrollen mit Daten und multilingualen Texten.

OnLoad überschreibt die geerbte Form-Methode und wird nur einmal aufgerufen, bevor die Dialogform angezeigt wird.

Die Form wird wegen Problemen mit FormStartPosition.CenterParent in InitializeComponent hier manuell zentriert.

**Kodierung 14-11: OnLoad für die Sprachedialogform**

In OnClosed wird die ausgewählte Sprache an das Datenfeld Language zugewiesen.

OnClosed überschreibt die geerbte Form-Methode und wird nur einmal aufgerufen, nachdem die Dialogform geschlossen wurde.

**Kodierung 14-12: OnClosed für die Sprachedialogform**

### 14.5.2 Die Dialogform zur Anzeige des Lizenzvertrages (EULA)

Die LIZENZ_DIALOG-Klasse erzeugt eine Dialogform zur Anzeige und Akzeptanz des Lizenzvertrages (EULA).

**Kodierung 14-13: Die LIZENZ_DIALOG-Klasse**

```
///*************************************************************************
/// Version 4.0 vom 01.04.18
///*************************************************************************
[CLSCompliantAttribute(true)]    // markiert die Klasse als CLS-kompatibel
protected class LIZENZ_DIALOG : Form
{

   #region Datenfelder, Konstruktor mit den Lizenztexten und InitializeComponent #endregion
   #region Bearbeitung der Ereignisse für die Dialogform und die Kontrollen #endregion

}
```

#### 14.5.2.1 Die Instanzvariablen

**Kodierung 14-14: Die Instanzvariablen**

#### 14.5.2.2 Der Konstruktor

Der Konstruktor für die LIZENZ_DIALOG-Klasse initialisiert ein zweidimensionales Array mit multilingualen Texten für die Kontrollen und erstellt die Lizenz-Dialogform in InitializeComponent.

**Kodierung 14-15: Der Konstruktor**

#### 14.5.2.3 InitializeComponent

In InitializeComponent wird die Lizenz-Dialogform mit den Kontrollen aufgebaut. Die Codierung wurde manuell in deutscher Sprache durchgeführt.

Der Aufruf erfolgt im Konstruktor.

**Kodierung 14-16: InitializeComponent**

#### 14.5.2.4 Bearbeitung der Ereignisse für die Lizenzdialogform und die Kontrollen

OnLoad initialisiert die Kontrollen mit Daten und multilingualen Texten.

OnLoad überschreibt die geerbte Form-Methode und wird nur einmal aufgerufen, bevor die Dialogform angezeigt wird.

Die Form wird wegen Problemen mit FormStartPosition.CenterParent in InitializeComponent hier manuell zentriert.

**Kodierung 14-17: OnLoad für die Lizenzdialogform**

PanelOnPaint gibt den sprachabhängigen Lizenztext zum Panel aus.

PanelOnPaint ist der eingetragene Handler für das Paint-Ereignis des Panels aus InitializeComponent.

**Kodierung 14-18: PanelOnPaint für die Lizenzdialogform**

### 14.5.3 Die Dialogform zur Passworteingabe

Die PASSWORT_DIALOG-Klasse erzeugt eine Dialogform zur Passworteingabe.

**Kodierung 14-19: Die PASSWORT_DIALOG-Klasse**

```
///*************************************************************************
/// Version 4.0 vom 01.01.18
///*************************************************************************
[CLSCompliantAttribute(true)]    // markiert die Klasse als CLS-kompatibel
protected class PASSWORT_DIALOG : Form
{

    #region Datenfelder, Konstruktor und InitializeComponent #endregion
    #region Bearbeitung der Ereignisse für die Dialogform und die Kontrollen #endregion
```

```
}
```

#### 14.5.3.1 Die Instanzvariablen

**Kodierung 14-20: Die Instanzvariablen**

#### 14.5.3.2 Der Konstruktor

Der Konstruktor für die PASSWORT_DIALOG-Klasse initialisiert ein zweidimensionales Array mit multilingualen Texten für die Kontrollen und erstellt die Passwort-Dialogform in InitializeComponent.

**Kodierung 14-21: Der Konstruktor**

#### 14.5.3.3 InitializeComponent

In InitializeComponent wird die Passwort-Dialogform mit den Kontrollen aufgebaut. Die Codierung wurde manuell in deutscher Sprache durchgeführt.

Der Aufruf erfolgt im Konstruktor.

**Kodierung 14-22: InitializeComponent**

#### 14.5.3.4 Bearbeitung der Ereignisse für die Passwortdialogform und die Kontrollen

OnLoad initialisiert die Kontrollen mit Daten und multilingualen Texten.

Lautet das zu kontrollierende Passwort "Testversion", wird "Testversion" in die Passwort-Textbox eingetragen.

OnLoad überschreibt die geerbte Form-Methode und wird nur einmal aufgerufen, bevor die Dialogform angezeigt wird.

Die Form wird wegen Problemen mit FormStartPosition.CenterParent in InitializeComponent hier manuell zentriert.

**Kodierung 14-23: OnLoad für die Passwortdialogform**

PasswortDialogPanelOnPaint gibt einen sprachabhängigen Erläuterungstext zum Panel aus.

PasswortDialogPanelOnPaint ist der eingetragene Handler für das Paint-Ereignis des Panels aus InitializeComponent.

**Kodierung 14-24: PasswortDialogPanelOnPaint für die Passwortdialogform**

OnClosed kontrolliert das eingegebene Passwort mit dem Kontrollpasswort unter Berücksichtigung des Start- und Ende-Index.

OnClosed überschreibt die geerbte Form-Methode und wird nur einmal aufgerufen, nachdem die Dialogform geschlossen wurde.

**Kodierung 14-25: OnClosed für die Passwortdialogform**

## 14.5.4 Die Dialogform zum Kopieren des Registrierungsschlüssels

Die REGISTRIERUNGS_DIALOG-Klasse erzeugt eine Dialogform zum Kopieren des ausgewählten Registrierungsschlüssels in die Zwischenablage.

**Kodierung 14-26: Die REGISTRIERUNGS_DIALOG-Klasse**

```
///*************************************************************************
/// Version 2.0 vom 12.10.09
///*************************************************************************
[CLSCompliantAttribute(true)]    // markiert die Klasse als CLS-kompatibel
protected class REGISTRIERUNGS_DIALOG : Form
{

    #region Datenfelder, Konstruktor und InitializeComponent #endregion
    #region Bearbeitung der Ereignisse für die Dialogform und die Kontrollen #endregion

}
```

### 14.5.4.1 Die Instanzvariablen

**Kodierung 14-27: Die Instanzvariablen**

### 14.5.4.2 Der Konstruktor

Der Konstruktor für die REGISTRIERUNGS_DIALOG-Klasse initialisiert ein zweidimensionales Array mit multilingualen Texten für die Form und erstellt die Registrierungs-Dialogform in InitializeComponent.

**Kodierung 14-28: Der Konstruktor**

### 14.5.4.3 InitializeComponent

In InitializeComponent wird die Registrierungs-Dialogform mit den Kontrollen aufgebaut. Die Codierung wurde manuell in deutscher Sprache durchgeführt.

Der Aufruf erfolgt im Konstruktor.

**Kodierung 14-29: InitializeComponent**

### 14.5.4.4 Bearbeitung der Ereignisse für die Registrierungsdialogform und die Kontrollen

OnLoad initialisiert die Kontrollen mit Daten und multilingualen Texten.

OnLoad überschreibt die geerbte Form-Methode und wird nur einmal aufgerufen, bevor die Dialogform angezeigt wird.

Die Form wird wegen Problemen mit FormStartPosition.CenterParent in InitializeComponent hier manuell zentriert.

**Kodierung 14-30: OnLoad für die Registrierungsdialogform**

RegistrierungsDialogPanelOnPaint gibt einen sprachabhängigen Erläuterungstext zum Panel aus. Hierzu wird auch die Position der Bildlaufleisten abgefragt.

RegistrierungsDialogPanelOnPaint ist der eingetragene Handler für das Paint-Ereignis des Panels aus InitializeComponent.

**Kodierung 14-31: RegistrierungsDialogPanelOnPaint für die Registrierungsdialogform**

Copy_Button_Click kopiert den Registrierungsschlüssel von dem Label in die Zwischenablage.

Der Aufruf erfolgt als eingetragener Handler für das Click-Ereignis des Copy-Button (Accept-Button) aus InitializeComponent.

**Kodierung 14-32: Copy_Button_Click für die Registrierungsdialogform**

In RB_RadioButton_Click wird nach Auswahl einer neuen Registrierungsart der Labeltext zum zugehörigen Registrierungsschlüssel gesetzt.

Der Aufruf erfolgt als eingetragener Handler für das Click-Ereignis der Radio-Buttons aus InitializeComponent.

**Kodierung 14-33: RB_RadioButton_Click für die Registrierungsdialogform**

### 14.5.5 Die Dialogform zur Eingabe des Aktivierungsschlüssels

Die AKTIVIERUNGS_DIALOG-Klasse erzeugt eine Dialogform zur Eingabe des Aktivierungsschlüssels.

Diese Dialogform wird innerhalb des Lizenzschutzablaufes nicht eingesetzt, weil der Aktivierungsschlüssel in einer Lizenzdatei enthalten ist, die an den Kunden übermittelt wird.

## 14.6 Der Mirage-Lizenzschutz

Die Methode ProtectMe von der Mirage Computer Systems GmbH wird z.T. korrigiert innerhalb der LIZENZSCHUTZ-Klasse des Basisprojekts zur Verfügung gestellt und von der Anwendung 1 und dem Formen-Verwaltungsprogramm in den Programmablauf eingebunden.

Der Mirage-Lizenzschutz wird durchgeführt, wenn das Symbol MirageLP für Bedingte Kompilierung den Eigenschaften für das Erstellen der Projekte zugefügt ist.

Der Aufruf der Kontrollmethode, in der ProtectMe() aufgerufen wird, erfolgt durch die Anwendung 1 in der Programmschleife des Basisprojekts und durch Multimedia Theater in der Kontrollmethode.

**Kodierung 14-34: Mirage Lizenzschutz-Methode ProtectMe**

# 15 Auf Sensoren zugreifen

Zur Erfassung von Sensordaten wird das Windows API Code Pack eingesetzt.

Hierzu müssen dem Visual Studio-Projekt zwei Verweise auf die benötigten dll-Dateien des Windows API Code Pack hinzugefügt werden: Microsoft.WindowsAPI.CodePack und Microsoft.WindowsAPICodePack.Sensors.

Des Weiteren ist in den Projekteigenschaften unter „Erstellen" das Symbol für Bedingte Compilierung „WAPICP" einzutragen.

Die Methoden für den Zugriff auf Sensoren sind in der Quellcodedatei WAPI_CodePack.cs enthalten, mit der die BASISPROJEKT-Klasse um das Windows API Code Pack erweitert wird.

**Hinweis 14 zum Windows API Code Pack**

Die im Windows API Code Pack 1.1 integrierten Klassen entsprechen nicht der Common Language Spezification (CLS)!

## 15.1 Die Sensordaten

Die Felder für die Sensoren werden mit der Instanz der BASISPROJEKT-Klasse erzeugt.

**Kodierung 15-1: Felder für die Sensoren**

## 15.2 Den Beschleunigungssensor einrichten

BS_Sensor_Init ist die Einrichtungsmethode für den Beschleunigungssensor.

Das Abfrageintervall des Beschleunigungssensors wird auf das kleinstmögliche gesetzt.

Zur Weitergabe neuer Sensordaten wird eine Ereignismethode eingerichtet.

Wenn der Beschleunigungssensor korrekt eingerichtet ist, gibt die Methode true zurück.

**Kodierung 15-2: BSSensor_Init**

## 15.3 Die Ereignismethode für den Beschleunigungssensor

BSSensor_Handler ist die Ereignismethode für den Beschleunigungssensor, wenn neue Daten vorliegen.

Wenn der Beschleunigungssensor gültig ist, werden die aktuellen Daten dem Beschleunigungsvektor übergeben.

Der Ereignishandler wird in BSSensor_Init erzeugt.

Für MULTITHREAD wird vor der Veränderung von öffentlichen Instanzvariablen der Bedienerthread mit dem Arbeitsthread synchronisiert.

**Kodierung 15-3: Ereignismethode BSSensor_Handler**

## 15.4 Den Beschleunigungssensor freigeben

Freigabemethode für den Beschleunigungssensor.

Der Eventhandler wird entfernt und der Beschleunigungssensor zu null gesetzt.

**Kodierung 15-4: BSSensor_Exit**

## 15.5 Den Umgebungslichtsensor einrichten

BS_Sensor_Init ist die Einrichtungsmethode für den Umgebungslichtsensor.

Das Abfrageintervall des Umgebungslichtsensors wird auf das kleinstmögliche gesetzt.

Zur Weitergabe neuer Sensordaten wird eine Ereignismethode eingerichtet.

Wenn der Umgebungslichtsensor korrekt eingerichtet ist, gibt die Methode true zurück.

**Kodierung 15-5: Lichtsensor_Init**

## 15.6 Die Ereignismethode für den Umgebungslichtsensor

Lichtsensor_Handler ist die Ereignismethode für den Umgebungslichtsensor, wenn neue Daten vorliegen.

Wenn der Umgebungslichtsensor gültig ist, wird die aktuelle Lichtstaerke in ein Feld abgespeichert.

Der Ereignishandler wird in Lichtsensor_Init erzeugt.

Für MULTITHREAD wird vor der Veränderung von öffentlichen Instanzvariablen der Bedienerthread mit dem Arbeitsthread synchronisiert.

**Kodierung 15-6:  Ereignismethode Lichtsensor_Handler**

## 15.7 Den Umgebungslichtsensor freigeben

Freigabemethode für den Umgebungslichtsensor.

Der Eventhandler wird entfernt und der Umgebungslichtsensor zu null gesetzt.

**Kodierung 15-7: Lichtsensor_Exit**

# 16 Eine Beispielanwendung

## 16.1 Die Klassen der Anwendung 1

Das Projekt Anwendung1 besteht aus den Klassen ANW1MAINCLASS, ANWENDUNG1, ANW1_GLOBALS, ANW1_TEXTE, ANW1_MENUE_TEXTE, EVENT_PROCS für die erweiterte Ereignisbearbeitung und den Klassen für die Dialogform ANW1_DIALOG, ANW1_DIALOG_TEXTE.

Alle Klassen sind in dem Namensraum Anw1 aufgeführt.

## 16.2 Die ANW1MAINCLASS-Klasse

ANW1MAINCLASS enthält neben der statischen Methode Main, die durch den Compiler als Startobjekt für die Anwendung aufgerufen wird, noch zwei Methoden als Handler von globalen Ausnahmen.

Für einen kontrollierten Ablauf, wird in der statischen Main-Methode in einem try-catch-Block ein ANWENDUNG1-Objekt mit einer separaten Anweisung erzeugt und die Standardmeldungsschleife für die Anwendung in einer zweiten Anweisung mit Application.Run gestartet, wodurch auch die Form sichtbar und bedienbar wird. Nach einer hierin abgefangenen und protokollierten Ausnahme wird die Anwendung beendet.

Zum globalen Abfangen und Protokollieren von unbehandelten Ausnahmen werden deshalb zwei Ausnahmehandler erzeugt, nach deren Aktivierung die Anwendung fortgeführt werden kann.

**Kodierung 16-1: ANW1MAINCLASS**

## 16.3 Die ANWENDUNG1-Klasse

Die Klasse ANWENDUNG1 wird von der BASISPROJEKT-Klasse abgeleitet und realisiert ein Anwendungsbeispiel. Sie enthält neben dem Konstruktor auch override-Methoden, die virtuelle Methoden des Basisprojekts überschreiben.

Das Programm der Anwendung 1 gibt wahlfrei einfache geometrische Figuren auf den Raumboden der Hintergrundgrafik aus. Hierzu wird die virtuelle Methode der BASISPROJEKT-Klasse überschrieben.

Zusätzlich zu den geerbten Fähigkeiten des Basisprogramms wird nach jeder Größenänderung der Form auf PanelOnResize eine Raumgrafik in die Hintergrundbitmap gezeichnet, die auf PanelOnPaint in das Grafikobjekt für die Ausgabe zur Client Area kopiert wird.

Des Weiteren erweitert die Anwendung das Basismenü und stellt eine eigene Dialogform mit zwei Seiten zur Verfügung. Die Raumgrafik kann über das Menü und auch über die Dialogform gewechselt werden. Zur Auswahl stehen drei verschiedene Muster, durchsichtig und einfarbig.

Die Klasse ANWENDUNG1 ist von der BASISPROJEKT-Klasse abgeleitet und realisiert ein Anwendungsbeispiel.

Diese Klassendefinition muss an erster Stelle des Quellcodes stehen, sonst ist keine Designer-Unterstützung der Form möglich.

**Kodierung 16-2: ANWENDUNG1-Klasse**

```
#region ANWENDUNG1-Klasse
///**********************************************************************
```

```
/// Version 4.0 vom 01.01.18
///*****************************************************************************
[CLSCompliantAttribute(true)]
public class ANWENDUNG1 : jkBPrj.BASISPROJEKT
{

    #region Datenfelder, Konstruktor, Programm, Programm_Init, Titeltext und
             InitializeComponent #endregion

    #region Erweiterung der Basismenüs #endregion

    #region Handler für die Form-Ereignisse #endregion

    #region Handler für die Menü-Ereignisse #endregion

    #region Handler für die Ereignisse vom Lichtsensor #endregion

}
#endregion
```

## 16.3.1 Die Instanzvariablen der Anwendung 1

Auf die implizit privaten Instanzvariablen kann nur innerhalb der ANWENDUNG1-Klasse zugegriffen werden. Einzig public sind die Instanzen von ANW1_GLOBALS für einen Zugriff auf die öffentlichen Datenfelder und ANW1_DIALOG für die Dialogform der Anwendung1 zum Löschen der Referenz nach dem Schließen der Dialogform.

**Kodierung 16-3: Die Instanzvariablen der ANWENDUNG1-Klasse**

## 16.3.2 Der Konstruktor der ANWENDUNG1-Klasse

Im ANWENDUNG1-Konstruktor werden Instanzen von den Basisklassen ANW1_GLOBALS und EVENT_PROCS erzeugt. Der Aufruf erfolgt in ANW1MAINCLASS.Main.

Dem Konstruktor der Basis wird für einen Kontrollablauf mit Passwort- oder Aktivierungs-schlüsselabfrage ein Programmpasswort übergeben. Bei einer Passwortabfrage wird "Testversion" als gültiges Passwort eingetragen. Für die Abfrage eines Aktivierungsschlüssels muss der dem Passwort folgende Parameter true sein.

Das MDI-Projekt übergibt dem Konstruktor keinen Parameter, weil der Kontrollablauf im Formen-Verwaltungsprogramm durchgeführt wird.

Der Lizenzschutz des mirage Licence Protector wird eingebunden, wenn der erste Parameter bool ist. Der zweite string-Parameter ist hierzu der Project Secure Key (PSK). Zur Kompilierung der hierfür erforderlichen Methoden und Variablen muss die Bedingte Kompilierungskonstante MirageLP definiert sein.

**Kodierung 16-4: Der Konstruktor der ANWENDUNG1-Klasse**

## 16.3.3 Der Titeltext für die Anwendung 1

Titeltext schreibt für die Anwendung 1 den mehrsprachigen Text in die Titelleiste der Form. Für das MDI-Projekt wird dem Programmtitel die Programmnummer vorangesetzt.

Titeltext überschreibt die geerbte Methode des Basisprojekts. Der Aufruf erfolgt durch das Basisprogramm, die Anwendung und das MDI-Programm.

**Kodierung 16-5: Titeltext für die Anwendung 1**

### 16.3.4 InitializeComponent für die Anwendung 1

InizializeComponent überschreibt die geerbte Methode des Basisprojekts und wird durch das Basisprogramm aufgerufen.

Weil die Form der Anwendung 1 mit den Menüs manuell codiert wird, darf der Inhalt dieser Methode mit dem Code-Editor verändert werden. Auf die Designerunterstützung wird wegen weitergehender Probleme verzichtet.

**Kodierung 16-6: InitializeComponent für die Anwendung 1**

## 16.4 Der Ablauf der Anwendung 1

### 16.4.1 Der Programmstart

Nach dem Programmstart wird OnLoad von der Anwendung ausgeführt und darin OnLoad des Basisprojekts aufgerufen. In OnLoad für das Basisprojekt wird nur für MULTITHREAD ein Arbeitsthread mit MT_Programmschleife als Startmethode und nicht für das MDI_PRJ ein Zeitgeber zur Aktivierung von Programmschleife gestartet.

In Programmscheife wird für den Ablauf mit MULTITHREAD nur ein erforderlicher Lizenzschutz-Kontrollablauf durchgeführt. Der eigentliche Programmablauf wird durch den Arbeitsthread in MT_Programmschleife ausgeführt, worin die Basis-Methode ständig aktiviert wird. In der Basis-Methode wird die von der Anwendung überschriebene Methode Programm virtuell aufgerufen.

### 16.4.2 Die Initialisierung

Programm_Init liest oder schreibt relevante öffentliche Variable des Basisprogramms und von dieser Anwendung 1 in eine Initialisierungsdatei.

Hierzu wird die virtuelle Methode des Basisprojekts überschrieben und die Basismethode selbst aufgerufen. Der Aufruf erfolgt innerhalb des Basisprogramms.

**Kodierung 16-7: Programm_Init der Anwendung 1**

### 16.4.3 Das Programm

Die Anwendung 1 gibt wahlfrei einfache geometrische Figuren auf den Raumboden der Hintergrundgrafik aus. Hierzu wird die virtuelle Methode von BASISPROJEKT überschrieben.

Der Aufruf erfolgt in BASISPROJEKT.Basis nur, wenn keine Programmpause.

**Kodierung 16-8: Programm von der Anwendung 1**

## 16.5 Erweiterungen und Änderungen zur Version 4.0

- Für die automatische Größenänderung von Steuerelementen für HighDpi-Anzeigen wird in der Projektdatei App.config der Zusatz <appSettings><add key="EnableWindowsFormsHighDpiAutoResizing" value="true" /></appSettings> Zugefügt.

- Das Abspielen von Audio- und Videodateien wird mit der AudioVideoPlayBack-API von Managed DirectX realisiert. Hierfür wird die Bedingte Kompilierungskonstante DAUDIOVIDEO angegeben. Für das Projekt muss als Zielplattform „x86" ausgewählt werden. Das Programm ist somit nicht mehr durch „Any CPU" plattformneutral. (s.a. Die Managed DirectX Bibliothek in das Projekt einbinden).
- Wenn eine Bilddatei angezeigt oder eine Video- oder Audiodatei abgespielt wird (Multimedia_Anzeige true), werden dafür in ANWENDUNG1.MenuCompleteHandler durch den Aufruf der Basismethode, der Dateiname und der Tooltiptext in der Statusbar angezeigt. Deshalb muss nach dem Programmstart auf OnLoad() und bei der Menuerzeugung in ErzeugeFormMenues() der Statusleistentext nicht mehr initialisiert werden.
- In ANWENDUNG1.Programm_Init wird die Konstante ANW1DATENENDE, wegen der Erhöhung der Konstanten BASISDATENENDE in BASISPROJEKT.Programm_Init auf 200, auf 300 erhöht.

## 16.5.1 Codeanalyse mit Visual Studio Community

Von der Codeanalyse für den ausgeführten Regelsatz „Alle Microsoft Regeln" werden viele Warnungen angezeigt, die bisher noch nicht alle beseitigt wurden.

Der ausgeführte Regelsatz kann unter den Projekteigenschaften\Codeanalyse ausgewählt werden. Von dem Regelsatz „Alle Microsoft Regeln" ausgehend, wird über den Öffnen-Taster durch Visual Studio eine neue Seite mit einer Liste von Warnungen angezeigt, die über Checkmarken abgewählt werden können.

Danach steht für die Anwendung1 ein eigener Regelsatz zur Auswahl.

Die grundlegend behobenen und abgewählten Regeln werden nur im Basisprojekt aufgelistet.

- Folgende Warnungen wurden behoben:
  CA1806: In Anw1_Dialog.cs müssen für Editierfelder die Text.Trim()-Anweisungen einem Datenfeld zugewiesen werden. Diese Anweisungen werden aber auch durch Convert.ToInt32() realisiert und können deshalb entfernt werden.
  CA2210: Assemblys müssen gültige starke Namen aufweisen:
  anhand der Fehlerbeschreibung wurde die Schlüsseldatei Anwendung1.snk erstellt.

## 16.5.2 Erweiterungen und Änderungen zur Version 3.0

- Die Raumgrafik wird mit dem Sensorwert für das Umgebungslicht gezeichnet (wenn vorhanden). Hierzu wird die Grafik-Dialogseite um eine Editierkontrolle für das Umgebungslicht mit Sensortaste erweitert.
- Zur Realisierung der Raumgrafik mit oder ohne Farbwechsel wird die Methode Groessen() mit dem Parameter Farbwechsel aufgerufen.
- Für die Methoden Programm, Titeltext und InitializeComponent wird der Zugriffsmodifikator wie im Basisprojekt von public zu protected geändert.
- In Programm_Init() werden wegen einer möglichen Ausnahme die Instanzen von BReader und BWriter innerhalb eines finally-blocks geschlossen.

## 16.5.3 Erweiterungen und Änderungen zur Version 2.0

- Zur weitergehenden Anzeige im Objektkatalog und für die kontextsensitive Hilfe den XML-Kommentar für alle Methoden mit <para>-Tags innerhalb der <summary>- und <remarks>-Tags versehen.
- Neben der Erhöhung der globalen Variablen catch_Ausnahmen, werden die innerhalb des Programmablaufes auftretenden Ausnahmen mit Datum und Uhrzeit, durch den Aufruf von

Ausnahmeprotokollierung() der BASIS_PROCS-Klasse in die Datei „Anwendung 1_Exceptions.txt" geschrieben.

- Zum globalen Abfangen und Protokollieren von unbehandelten Ausnahmen werden in ANW1MAINCLASS.Main zwei Ausnahmehandler erzeugt, nach deren Aktivierung die Anwendung fortgeführt wird.
- Die ANW1_DIALOG-Klasse erhält neue Methoden zur Initialisierung der Kontroll- und Textwerte der einzelnen Dialogseiten aus dem Programmablauf des Arbeitsthreads heraus.
- Zur Initialisierung nach dem Programmstart für MDI-Programmformen und zur Anpassung an eine neue Sprachauswahl, wird in ErzeugeFormMenues() eine geöffnete Anwendungsdialogform durch Aufruf ihres Load-Handlers neu aufgebaut.
- Aufgrund von Änderungen im Basisprojekt, wird die OnResize-Methode der Form zu dem in ErzeugeFormPanel() eingetragenen Handler PanelOnResize() für das Resize-Ereignis des Panels geändert. PanelOnResize() wird über das Basisprojekt in OnResize() nur aktiviert, wenn die Form nicht zum Symbol verkleinert wird oder vom Symbol wieder hergestellt wurde.
- Aufruf von PanelOnResize() in RaumgrafikmenueOnClick().
- In EPPOCS.Groessen() wird nicht mehr der FormWindowState abgefragt und der gesamte Ablauf mit dem Arbeitsthread synchronisiert. Durch FormPanel.Invalidate() wird zur Anzeige des Hintergrundes ein PanelOnPaint-Ereignis angestoßen.
- Zur Anpassung an eine neue Sprachauswahl wird in ErzeugeFormMenues() eine geöffnete Anwendungsdialogform durch Aufruf ihres Load-Handlers neu aufgebaut.
- Der AboutBox werden vor der Anzeige eine neue Imagedatei und zwei LinkLabels zugefügt. Dies gilt nur für die Anzeige über den Menüpunkt „Hilfe/über...". Nach der Anzeige durch den „über"-Button der Basis-Dialogseite wird derzeit nur der Beschreibungstext neu zugefügt.
- Im Konstruktor der Dialogformen werden für die unter .Net 2.0 empfohlenen Ereignisse FormClosing und FormClosed die Handler ANWENDUNG_DIALOG_FormClosed und ANWENDUNG_DIALOG_FormClosing eingetragen. In ANWENDUNG_DIALOG_FormClosing wird die Dialogform nicht nach dem Schließen einer Programmform, wenn diese nicht die letzte Form ist, sondern nur nach Betätigung eines Schließen-Button geschlossen.
- In KontextMenueOnClick für die Anwendung 1 wird für das MDI-Projekt die Dialogform nur angezeigt, wenn nicht schon eine für eine andere MDI-Form geöffnet wurde. Hierzu wird die schon vorhandene Kodierung aus ProgrammMenueOnClick übernommen.

## 16.6 Die öffentlichen Datenfelder der Anwendung 1

Datenfelder, die für andere Klassen öffentlich zugänglich sein müssen, werden in einer eigenen Klasse ANW1_GLOBALS aufgeführt. Auf diese Datenfelder kann hierdurch über eine separate Referenz zugegriffen werden.

Der Konstruktor von ANW1_GLOBALS initialisiert die öffentlichen Datenfelder der Anwendung 1.

Die Instanz von ANW1_GLOBALS wird im Konstruktor der ANWENDUNG1-Klasse erzeugt.

Von ANW1_GLOBALS werden automatisch Objekte von ANW1_TEXTE und ANW1_MENUE_TEXTE als Basisklassen abgeleitet. Diese enthalten statische, zwei- oder dreidimensionale Arrays in denen die mehrsprachigen Texte für die Anwendung und das Menü eingetragen werden.

Die Menütexte enthalten zusätzlich einen kleinen Erläuterungstext zur Menüauswahl, der für die Ausgabe zur Statusleiste verwendet wird.

**Kodierung 16-9: die ANW1_GLOBALS-Klasse**

### 16.6.1 Die mehrsprachigen allgemeinen Texte der Anwendung 1

Die ANW1_TEXTE-Klasse enthält mehrdimensionale, öffentlich zugängliche Textarrays für multilinguale Projekt- und Messagebox-Texte der Anwendung 1 und erbt die Menütexte von ANW1_MENUE_TEXTE. Eine Instanz wird mit der ANW1_GLOBALS-Klasse erzeugt.

Im Konstruktor werden die multilingualen Texte für die Anwendung 1 in den Textarrays bereitgestellt.

Zur Seiteneinsparung werden nur die deutschen und englischen Texte aufgeführt.

**Kodierung 16-10: Die ANW1_TEXTE-Klasse**

### 16.6.2 Die mehrsprachigen Menütexte der Anwendung 1

Die ANW1_MENUE_TEXTE-Klasse enthält mehrdimensionale, öffentlich zugängliche Textarrays für die multilingualen Menütexte der Anwendung 1 mit Erläuterungstexten. Eine Instanz wird mit der ANW1_TEXTE-Klasse erzeugt.

Im Konstruktor werden die multilingualen Menütexte für die Anwendung 1 in den Textarrays bereitgestellt.

Zur Seiteneinsparung werden nur die deutschen und englischen Texte aufgeführt.

**Kodierung 16-11: Die ANW1_MENUE_TEXTE-Klasse**

## 16.7 Erweiterung der Basismenüs

In ErzeugeFormMenues wird das multilinguale Haupt- und Kontextmenü des Basisprojekts für die Anwendung 1 erweitert. Hierzu wird die gleichnamige virtuelle Basismethode überschrieben und selbst aufgerufen. Danach werden die Menüpunkte verändert und erweitert.

Eine geöffnete Anwendungsdialogform wird zur Anpassung an eine neue Sprachauswahl durch Aufruf ihres Load-Handlers neu aufgebaut.

Die Aufrufe erfolgen innerhalb des Basisprojekts in OnLoad, SpracheMenueOnClick und HilfeTextSpracheMenueOnClick.

Nur für MULTITHREAD gilt: wenn diese Methode nicht von ihrem Erzeuger-Thread aufgerufen wird, wird sie durch ihre Delegat-Methode im Erzeuger-Thread ausgeführt.

**Kodierung 16-12: ErzeugeFormMenues für die Anwendung 1**

## 16.8 Erweiterung des Kontextmenüs

ErzeugeKontextMenue erweitert das Basis-Kontextmenü für die Anwendung 1. Hierzu wird die gleichnamige virtuelle Basismethode überschrieben und selbst aufgerufen.

Der Aufruf erfolgt innerhalb des Basisprojekts in ErzeugeFormMenues.

**Kodierung 16-13: ErzeugeKontextMenue für die Anwendung 1**

# 16.9 Form-Ereignisse für die Anwendung 1

## 16.9.1 OnLoad

OnLoad Initialisiert die Form und die Anwendung1. Hierzu wird die überschriebene Basismethode aufgerufen.

OnLoad wird nur einmal als Handler für das Load-Ereignis aufgerufen, bevor die Form angezeigt wird.

In der Basismethode werden die Menüs und die StatusBar- und Panel-Kontrolle erzeugt und zur Initialisierung die virtuellen Methoden Programm_Init, InitializeComponent und Titeltext aufgerufen.

Nur für MULTITHREAD wird ein Arbeitsthread mit MT_Programmschleife als Startmethode und nicht für das MDI_PRJ ein Zeitgeber zur Aktivierung von Programmschleife gestartet.

**Kodierung 16-14: OnLoad für die Anwendung 1**

## 16.9.2 PanelOnResize

PanelOnResize ruft die überschriebene Basismethode und für die erweiterte Ereignisbehandlung Groessen von der EVENT_PROCS-Klasse auf.

Der Aufruf erfolgt als Handler für das Resize-Ereignis des Panels und explizit auf OnLoad und nach einer Raumgrafikauswahl in RaumgrafikMenueOnClick.

Die Basismethode erzeugt zur allgemeinen Verwendung ein GDI+ Grafik-Objekt abgeleitet von dem Panel in der neuen Größe der Client Area des Panels. In dieser Größe wird auch eine Basisbitmap mit einem zugehörigen GDI+ Grafik-Objekt erzeugt.

**Kodierung 16-15: PanelOnResize für die Anwendung 1**

## 16.9.3 OnClosed

OnClosed gibt von der Anwendung erzeugte Objekte frei (wenn vorhanden) und ruft danach OnClosed von der Basis auf.

Der Aufruf erfolgt als Handler für das Closed-Ereignis.

**Kodierung 16-16: OnClosed für die Anwendung 1**

## 16.9.4 Dispose

Dispose gibt die von Form verwendeten Ressourcen (mit Ausnahme des Speichers) frei und ruft danach die überschriebene Basismethode der Form auf. Dispose wurde als Standardmethode unverändert übernommen.

Der Aufruf erfolgt nachdem OnClosed für die Anwendung 1 durchlaufen wurde.

**Kodierung 16-17: Dispose für die Anwendung 1**

## 16.10 Erweiterte Ereignisbearbeitung in der EVENT_PROCS-Klasse

Die EVENT_PROCS-Klasse enthält Methoden für die Anwendung 1 zur erweiterten Ereignisbearbeitung und zum Zeichnen der Raumgrafik.

**Kodierung 16-18: Die EVENT_PROCS-Klasse**

```
#region EVENT_PROCS-Klasse
///*************************************************************************
/// Version 4.0 vom 07.12.17
///*************************************************************************
[CLSCompliantAttribute(true)]    // markiert die Klasse als CLS-kompatibel
public class EVENT_PROCS
{

    #region Datenfelder und Konstruktor #endregion

    #region Erweiterte Ereignisbearbeitung #endregion

    #region Zeichnen der Raumgrafik in die Basisbitmap #endregion

}
#endregion
```

### 16.10.1 Die Instanzvariablen

Auf die implizit privaten Instanzvariablen kann nur innerhalb der EVENT_PROCS-Klasse zugegriffen werden.

**Kodierung 16-19: Die Datenfelder der EVENT_PROCS-Klasse**

### 16.10.2 Der Konstruktor

Als Parameter werden dem Konstruktor der EVENT_PROCS-Klasse die Referenzen auf die Instanzen der öffentlichen Datenfelder der Anwendung 1 und des Basisprojekts übergeben. Hierdurch wird in EVENT_PROCS der Zugriff auf das Basisprojekt und die Form ermöglicht.

Die Instanz der EVENT_PROCS-Klasse wird im Konstruktor der ANWENDUNG1-Klasse erzeugt.

**Kodierung 16-20: Der Konstruktor der EVENT_PROCS-Klasse**

### 16.10.3 Groessen auf PanelOnResize

Groessen berechnet wichtige Programmvariable und ruft Hintergrund auf, wenn keine Multimediadatei angezeigt wird.

Für MULTITHREAD wird der Ablauf zuvor mit dem Arbeitsthread synchronisiert.

Groessen wird nach jeder Ausführung von PanelOnResize aufgerufen und weitergehend mehrfach zur Aktualisierung der Raumgrafik.

In PanelOnResize wird bei einer Programmausnahme auch eine Ausnahmeprotokollierung durchgeführt.

**Kodierung 16-21: Groessen auf PanelOnResize für die Anwendung 1**

## 16.11 Zeichnen der Raumgrafik in die Basisbitmap

Hintergrund zeichnet die ausgewählte Raumgrafik in die vom Basisprojekt zur Verfügung gestellte Basisbitmap oder gibt eine neue Farbe für den Hintergrund der Form an. Der Aufruf erfolgt in Groessen.

**Kodierung 16-22: Hintergrund**

### 16.11.1 Raumfarben

Raumfarben zeichnet die Raumwände mit linear abnehmenden Farbwerten und den Boden mit der Hintergrundfarbe. Hierbei wird für alle Farbwerte durch diffuse Reflexion der Umgebungslichtwert berücksichtigt.

Der Aufruf erfolgt in Hintergrund.

**Kodierung 16-23: Raumfarben**

### 16.11.2 Hatchmuster

Hatchmuster zeichnet die Raumwände mit Hatch-Mustern. Der Aufruf erfolgt in Hintergrund.

**Kodierung 16-24: Hatchmuster**

### 16.11.3 LinearGradient_Farbmuster

LinearGradient_Farbmuster zeichnet die Raumwände mit dem LinearGradientBrush mit zwei Wandfarben und vier wechselnden Raumfarben.

Hierbei wird für alle Farbwerte durch diffuse Reflexion der Umgebungslichtwert berücksichtigt.

Der Aufruf erfolgt in Hintergrund.

**Kodierung 16-25: LinearGradient_Farbmuster**

### 16.11.4 PathGradientmuster

PathGradientmuster zeichnet die Raumwände und den Raumboden mit PathGradient-Mustern. Der Aufruf erfolgt in Hintergrund.

**Kodierung 16-26: PathGradientmuster**

### 16.11.5 Diffuse_Reflexion

Ausgehend von einem Farbwert in der Nähe berechnet Diffuse_Reflexion einen diffusen Farbwert in der Entfernung.

Hierbei wird auch der Umgebungslichtwert aus der Editierkontrolle der Grafikdialogseite oder von einem aktiven Sensor berücksichtigt.

**Kodierung 16-27: Diffuse_Reflexion**

## 16.12 Ereignisse von den Menüs der Anwendung 1

### 16.12.1 Das Abschließende Menüereignis

In MenuCompleteHandler wird zur Aktualisierung der Statusleiste für Multimedia_Anzeige true die Basismethode aufgerufen. Anderenfalls wird der Starttext der Anwendung 1 zum ersten Panel der StatusBar ausgegeben.

Der Aufruf erfolgt als überschreibender Handler für das MenuComplete-Ereignis der Form, nachdem die Menübehandlung abgeschlossen wurde und explizit auf OnMenuComplete im MDI-Verwaltungsprogramm.

**Kodierung 16-28: MenuCompleteHandler für die Anwendung 1**

### 16.12.2 Ereignisse für das Programmmenüelement

ProgrammMenueOnClick bearbeitet die Clicks der von der Anwendung 1 erzeugten Elemente des Programm-Menüs.

Bei Auswahl von "Dialog" wird die modeless Dialogform der Anwendung nur dann erzeugt und angezeigt, wenn nicht schon eine erzeugt wurde.

Auf Menüauswahl "Beenden !" wird die Anwendung beendet oder für das MDI-Projekt die Kindform geschlossen.

Der Aufruf erfolgt als ein in ErzeugeFormMenues für Click-Ereignisse eingetragener Handler, wenn ein Programm-Menüpunkt angeklickt wurde. Die gleichnamige Methode des Basisprojekts wird überschrieben. Diese wird selbst nur für den Pause-Menüindex aufgerufen. Der public-Zugriffsmodifizierer ist wegen direkter Methodenaufrufe erforderlich.

Nur für MULTITHREAD wird vorab geprüft, ob der Aufruf von dem Erzeuger-Thread erfolgt. Ist dies nicht der Fall, wird ProgrammMenueOnClick erneut durch ihre Delegat-Methode **synchron** im Erzeuger-Thread ausgeführt.

**Kodierung 16-29: ProgrammMenueOnClick für die Anwendung 1**

ProgrammMenueOnSelect gibt den Hilfetext für den ausgewählten Menüpunkt zum ersten Panel der StatusBar aus.

Der Aufruf erfolgt als ein in ErzeugeFormMenues für Select-Ereignisse eingetragener Handler, wenn ein Programm-Menüpunkt ausgewählt wurde. Die gleichnamige Methode des Basisprojekts wird überschrieben und nur für den Pause-Menüindex aufgerufen.

**Kodierung 16-30: ProgrammMenueOnSelect für die Anwendung 1**

### 16.12.3 Ereignisse für das Raumgrafikmenüelement

RaumgrafikMenueOnClick bearbeitet die Clicks der Raumgrafik-Menüelemente. Die Raumgrafik wird durch Aufruf von Groessen mit Farbwechsel  in die neue BasisBitmap gezeichnet.

Der Aufruf erfolgt als ein in ErzeugeFormMenues für Click-Ereignisse eingetragener Handler, wenn ein Raumgrafik-Menüelement angeklickt wurde und durch PerformClick, wenn eine Raumgrafik über die Dialogform ausgewählt wurde.

**Kodierung 16-31: RaumgrafikMenueOnClick für die Anwendung 1**

RaumgrafikMenueOnSelect gibt den Hilfetext für den ausgewählten Raumgrafik-Menüpunkt zum ersten Panel der StatusBar aus.

Der Aufruf erfolgt als ein in ErzeugeFormMenues für Select-Ereignisse eingetragener Handler, wenn ein Raumgrafik-Menüelement ausgewählt wurde.

**Kodierung 16-32: RaumgrafikMenueOnSelect für die Anwendung 1**

## 16.12.4 Ereignisse für das Kontextmenü

KontextMenueOnClick bearbeitet die Clicks der von der Anwendung 1 erzeugten Elemente des Kontextmenüs.

Bei Auswahl von "Dialog" wird die modeless Dialogform der Anwendung nur dann erzeugt und angezeigt, wenn nicht schon eine erzeugt wurde.

Der Aufruf erfolgt als ein in ErzeugeKontextMenue für Click-Ereignisse eingetragener Handler, wenn ein Kontextmenüpunkt angeklickt wurde. Die gleichnamige Methode des Basisprojekts wird überschrieben und selbst aufgerufen, wenn der Menüindex kleiner als sechs ist.

**Kodierung 16-33:  KontextMenueOnClick für die Anwendung 1**

KontextMenueOnSelect gibt den Hilfetext für den ausgewählten Kontextmenüpunkt zum ersten Panel der StatusBar aus.

Der Aufruf erfolgt als ein in ErzeugeKontextMenue für Select-Ereignisse eingetragener Handler, wenn ein Kontextmenüpunkt ausgewählt wurde. Die gleichnamige Methode des Basisprojekts wird überschrieben und nur aufgerufen, wenn der Menüindex kleiner als sechs ist.

**Kodierung 16-34: KontextMenueOnSelect für die Anwendung 1**

## 16.12.5 Ereignisse für das Hilfemenüelement

HilfeMenueOnClick bearbeitet die Clicks der für die Anwendung 1 relevanten Elemente des Hilfe-Menüs.

Nach Auswahl von "Einführung" und "Menübefehle" wird aus den multilingualen HTML-Hilfetextdateien "Anw1_4_0_?Language?.chm" ein Inhaltsverzeichnis mit dem Einführungstext oder der Beschreibung der Menübefehle aufgerufen.

Nach Auswahl von "über..." wird eine About-Dialogbox mit multilingualer Information über die Anwendung 1 angezeigt.

Der Aufruf erfolgt als ein in ErzeugeFormMenues für Click-Ereignisse eingetragener Handler, wenn ein Hilfemenüpunkt angeklickt wurde. Der gleichnamige Handler des Basisprojekts wird überschrieben.

**Kodierung 16-35: HilfeMenueOnClick für die Anwendung 1**

#### 16.12.5.1 Das Click-Ereignis für die LinkLabels

WebLinkLabel_LinkClicked bearbeitet die Clicks auf die LinkLabels innerhalb des Beschreibungstextes der AboutBox.

Der Aufruf erfolgt als ein in HilfeMenueOnClick für Click-Ereignisse von den LinkLabels eingetragener Handler.

**Kodierung 16-36: WebLinkLabel_LinkClicked**

## 16.13 Die Ereignismethode für den Umgebungslichtsensor

Lichtsensor_Handler ist die Ereignismethode für den Umgebungslichtsensor, wenn ein neuer Umgebungslichtwert vorliegt.

Die Raumgrafik für den neuen Sensorwert wird durch einen Aufruf von EVENT_PROCS.Groessen realisiert.

Für MULTITHREAD wird Lichtsensor_Handler erneut durch einen Delegaten im Erzeuger-Thread aufgerufen, wenn dies erforderlich ist.

Lichtsensor_Handler überschreibt die virtuelle Methode des Basisprojekts und ruft diese zu Beginn direkt auf.

**Kodierung 16-37:  Ereignismethode Lichtsensor_Handler**

# 17 Die Dialogform der Anwendung 1

Die Dialogform der Anwendung 1, die in der ANW1_DIALOG-Klasse durch Vererbung von der Form-Klasse erzeugt wird, besteht aus drei Seiten mit TabControls zur Seitenauswahl.

Auf der Programmseite kann der Ablauf mit der Pause / Start-Taste unterbrochen und wieder gestartet werden. In einem Textfeld wird die Anzahl der Aufrufe der Anwendung 1 in einer Sekunde ausgeben. Mit Betätigung der „über...“-Taste wird eine weitere Dialogform mit Informationen zur Programmversion angezeigt.

Von der Grafikseite kann über fünf Optionsfelder die Raumgrafik ausgewählt werden. Das Umgebungslicht kann durch ein Texteingabefeld und eine Sensortaste verändert werden.

Die Grafik 2-Seite enthält vorerst keine Kontrollen.

**Kodierung 17-1: die ANW1_DIALOG-Klasse**

```
///*************************************************************************
/// Version 4.0 vom 07.01.17
///*************************************************************************
[CLSCompliantAttribute(true)]     // markiert die Klasse als CLS-kompatibel
public class ANW1_DIALOG : Form
{

    #region Datenfelder und Konstruktor #endregion

    #region InitializeComponent für den Visual Studio Designer #endregion

    #region Ereignisse zur Verwaltung der Dialogform #endregion

    #region Methoden und Ereignis-Handler für die Programm-Dialogseite und den Timer #endregion

    #region Methoden und Ereignis-Handler für die Grafik-Dialogseite #endregion

    #region Methoden und Ereignis-Handler die Grafik 2-Dialogseite #endregion

}
```

## 17.1 Die Instanzvariablen

Die Dialogkontrollen wurden von dem Designer bei der Erstellung der Dialogformen zugefügt und dürfen nicht manuell geändert werden.

**Kodierung 17-2: die Instanzvariablen der Anwendung 1-Dialogform**

## 17.2 Der Konstruktor

Im Konstruktor für die ANW1_DIALOG-Klasse werden die als Parameter übergebenen Referenzen als Instanzvariable abgespeichert. Danach wird eine Instanz von ANW1_DIALOG_TEXTE erstellt und in InitializeComponent die mit dem Designer erstellte Dialogform für die Anwendung 1 aufgebaut.

Der Aufruf erfolgt nach Menüpunktauswahl "Dialog..." in  ProgrammMenueOnClick und KontextMenueOnClick.

**Kodierung 17-3: der Konstruktor für die Dialogform der Anwendung 1**

## 17.3 Die Dialogform mit dem Visual Studio-Designer erzeugen

InitializeComponent erzeugt die Dialogseiten der Anwendung 1 mit den Kontrollen. Der Aufruf erfolgt im ANW1_DIALOG-Konstruktor.

Die Kodierung wird automatisch mit dem in Visual Studio integrierten Designer in deutscher Sprache erzeugt und darf nicht manuell geändert werden.

ANW1_DIALOG_HelpRequested ist als Handler für das HelpRequested-Ereignis der Dialogform eingetragen. OKButton_Click, AbbrechenButton_Click, HilfeButton_Click, PauseButton_Click, Info_Button_Click, RGRadioButton_Click sind die eingetragenen Handler für die Click-Ereignisse der Button-Kontrollen.

```
///*************************************************************************
/// Version 4.0 vom 27.03.18
/// Ein- und Ausgabedaten:
/// alle Kontrollen der Dialogform der Anwendung 1
///*************************************************************************
private void InitializeComponent()
{...}
```

### 17.3.1 Veränderungen am Design

Zur besseren Bedienung auf hochauflösenden Displays wurde für die Version 4.0 die Schrift für alle Dialogseiten auf Microsoft Sans Serif 10 (9,75 Pt) vergrößert (vorher 8,25Pt).

Hierzu wurde die spanische Beschreibung für die Abbrechen-Taste auf Cancellar verändert (Franz. Annuler).

Auf der Programm-Seite wurden der Pause / Start-Taster und die Aufrufe-Groupbox an die multilingualen Übersetzungstexte angepasst (die spanische Übersetzung für Aufrufe wurde zu Llamadas geändert). Die Über-Taste wurde leicht vergößert.

Auf der Grafik-Seite wurde die Umgebungslicht Groupbox verbreitert. Die Textbox für das Umgebungslicht wurde an die Position 16x25 verschoben und für die neue Fontgröße 11 (11,25 Pt) auf 77x24 vergrößert. Die inneren Drehfelder wurden auf 25x20 und der Sensor-Taster auf 77x30 angepasst.

## 17.4 Ereignisse zur Verwaltung der Dialogform

### 17.4.1 LoadHandler

LoadHandler initialisiert die Kontrollen mit Daten und multilingualen Texten und führt ToolTips und erweiterte Hilfetexte ein.

LoadHandler ist der im Konstruktor eingetragene Handler für das Load-Ereignis.

Die Aufrufe erfolgen bevor die Dialogform angezeigt wird und explizit zur Korrektur der Kontrollen nach einer Menüerzeugung in ErzeugeFormMenues und im MDI-Projekt nach einem Formwechsel in MdiChildActivateHandler.

Zur vollständigen Initialisierung der Anwendung 1-Dialogform werden die Init-Methoden der einzelnen Dialogseiten aufgerufen.

Nur für MULTITHREAD gilt, wenn diese Methode nicht von ihrem Erzeuger-Thread aufgerufen wird, wird sie durch ihre Delegat-Methode im Erzeuger-Thread ausgeführt.

**Kodierung 17-4: LoadHandler für die Anwendung 1-Dialogform**

## 17.4.2 Anw1_DIALOG_HelpRequested

Anw1_DIALOG_HelpRequested zeigt eine Message-Box mit dem ausführlichen Hilfetext der ausgewählten Kontrolle an. Der Aufruf erfolgt als ein im Designer eingetragener Handler für das HelpRequested-Ereignis der Basis-Dialogform nach Anwahl einer Kontrolle über den ?-Button oder die F1-Taste.

Das Ereignis wird auch ausgelöst, wenn der Kontrolle über die HelpProvider-Klasse ein Popup-Hilfetext zugewiesen wurde und dieser durch SetShowHelp für die Kontrolle deaktiviert ist.

**Kodierung 17-5: Anw1_DIALOG_HelpRequested für die Anwendung 1-Dialogform**

## 17.4.3 ANW1_DIALOG_FormClosing

Für die Einzelanwendung wird nur die für die ANWENDUNG 1 eingetragene Referenz auf das ANW1_DIALOG-Objekt zu null gesetzt. Für das MDI_PRJ werden bei einem Aufruf nach einer durch die Form geschlossenen Anwendung, für alle geöffneten Anwendungen die Referenzen zur Dialogform eingetragen, die Anzahl der Kindformen summiert und für die Dialogform ein neuer Eigner ermittelt.

Sind mehrere Kindformen vorhanden, wird das Close-Ereignis für die Form der Anwendung 1 und die Dialogform abgebrochen, der Dialogform der neue Anwendung 1-Eigner zugewiesen und dieser aktiviert. Abschließend wird das Close-Ereignis nochmals für den alten Eigner der Dialogform ausgelöst, wodurch diesmal nur die MDI-Kindform der Anwendung 1 geschlossen wird.

Bei einem Aufruf nach einem expliziten Schließen der Dialogform durch einen Button-Click, wird für das MDI_PRJ für alle Kindformen der Anwendung 1 die Referenz zur Dialogform zu null gesetzt.

ANW1_DIALOG_FormClosing wird als ein im Konstruktor eingetragener Handler für das FormClosing-Ereignis aufgerufen bevor die Dialogform geschlossen wird.

**Kodierung 17-6: ANW1_DIALOG_FormClosing für die Dialogform der Anwendung 1**

## 17.4.4 ANW1_DIALOG_FormClosed

Gibt von dieser Klasse erzeugte Objekte und das ANW1_DIALOG-Objekt selbst frei.

ANW1_DIALOG_FormClosed wird als ein im Konstruktor eingetragener Handler für das FormClosed-Ereignis aufgerufen bevor die Dialogform geschlossen wird.

**Kodierung 17-7: ANW1_DIALOG_FormClosed für die Dialogform der Anwendung 1**

## 17.4.5 OKButton_Click

OKButton_Click speichert den letzten Dialogseitenindex, setzt die Initialisierungsvariable false und schließt den Dialog mit Close.

Der Aufruf erfolgt als Handler für das Click-Ereignis des OK-Button und als Accept-Ereignis für den Dialog.

**Kodierung 17-8: OKButton_Click für die Anwendung 1-Dialogform**

### 17.4.6 AbbrechenButton_Click

In AbbrechenButton_Click werden die Einstellungen der Dialogform beim ersten Aufruf wieder hergestellt.

Für das MDI-Projekt werden für alle geöffneten und zumindest einmal aktivierten Formen die Abbrechenwerte wiederhergestellt.

Für MULTITHREAD wird vor der Veränderung der globalen Variablen der Bedienerthread mit dem Arbeitsthread synchronisiert.

Der Aufruf erfolgt als Handler für das Click-Ereignis des Abbrechen-Button und als Cancel-Ereignis für den Dialog.

**Kodierung 17-9: AbbrechenButton_Click für die Anwendung 1-Dialogform**

### 17.4.7 HilfeButton_Click

Ruft für die aktuelle Dialogseite die HTML-Hilfe auf.

Aufruf als Handler für das Click-Ereignis des Hilfe-Button.

**Kodierung 17-10: HilfeButton_Click für die Dialogform der Anwendung1**

## 17.5 Mehrsprachige Texte für die Dialogform

Die Dialogtexte werden mehrsprachig in der Klasse ANW1_DIALOG_TEXTE zur Verfügung gestellt. Sie bestehen aus dem Titeltext, den Texten für die TabControls und Messageboxtexten, die alle ohne Tooltips auskommen.

Die weiteren Texte für die Kontrollen aller Dialogseiten enthalten Tooltips und, wenn erforderlich, zusätzlich erweiterte Hilfetexte. Wenn ein erweiterter Hilfetext zur Verfügung steht, wird dem Tooltip ein Hinweis in der Form (-> ?-Hilfe) oder (-> Hilfetext) zugefügt.

Eine Instanz von ANW1_DIALOG_TEXTE wird im Konstruktor der ANW1_DIALOG-Klasse erzeugt.

Zur Seiteneinsparung werden nachfolgend nur die deutschen und englischen Texte aufgeführt.

**Kodierung 17-11: Die ANW1_DIALOG_TEXTE-Klasse**

## 17.6 Ereignisse für die Kontrollen der Programm-Dialogseite

### 17.6.1 Programm_Init

Programm_Init initialisiert die Kontrollen der Programm-Dialogseite mit Daten (Kontrollen_Init true) und multilingualen Texten und ToolTips (Text_Init true).

Mit Anw1Dialog_Init false, werden die Rückladewerte für den Abbruch der Dialogseite als globale Daten der Anwendung abgespeichert.

Des Weiteren wird ein Sekunden-Timer für den Eintrag der Aufrufe/s in den AufrufeLabel erzeugt.

Die Aufrufe erfolgen in LoadHandler und explizit im Programmablauf zur Aktualisierung der Kontrollen der Programm-Dialogseite.

Nur für MULTITHREAD gilt, wenn diese Methode nicht von ihrem Erzeuger-Thread aufgerufen wird, wird sie durch ihre Delegat-Methode im Erzeuger-Thread ausgeführt.

**Kodierung 17-12: Programm_Init für die Programm-Dialogseite**

### 17.6.2 Die Timer-Methode AufrufeSekTimer

Timer-Methode des Anw1DlgSekTimer für den Eintrag der Aufrufe/s in ein Label der Programm-Dialogseite. Der Timer wird mit einem Intervall von einer Sekunde in Programm_Init erzeugt.

**Kodierung 17-13:  Timer-Methode AufrufeSekTimer**

### 17.6.3 PauseButton_Click

Unterbricht oder Startet die Anwendung 1.

Der Aufruf erfolgt als Ereignis-Handler des "Pause / Start"-Button auf der Basis-Dialogseite.

**Kodierung 17-14: PauseButton_Click für die Anwendung 1-Dialogform**

### 17.6.4 InfoButton_Click

InfoButton_Click zeigt eine About-Dialogbox mit multilingualen Informationen über die Anwendung 1 an.

Der Aufruf erfolgt als Ereignis-Handler für den "über"...-Button der Basis-Dialogseite.

**Kodierung 17-15: InfoButton_Click für die Anwendung 1-Dialogform**

## 17.7 Ereignisse für die Kontrollen der Grafik-Dialogseite

### 17.7.1 Grafik_Init

Grafik_Init initialisiert die Kontrollen der Grafik-Dialogseite mit Daten (Kontrollen_Init true) und multilingualen Texten und ToolTips (Text_Init true).

Mit Anw1Dialog_Init false, werden die Rückladewerte für den Abbruch der Dialogseite als globale Daten der Anwendung abgespeichert.

Die Aufrufe erfolgen in LoadHandler und explizit im Programmablauf zur Aktualisierung der Kontrollen der Grafik-Dialogseite.

Nur für MULTITHREAD gilt, wenn diese Methode nicht von ihrem Erzeuger-Thread aufgerufen wird, wird sie durch ihre Delegat-Methode im Erzeuger-Thread ausgeführt.

**Kodierung 17-16: Grafik_Init für die Grafik-Dialogseite**

## 17.7.2 RgRadioButton_Click

RgRadioButton_Click bearbeitet eine neue Auswahl der Raumgrafik.

Der Aufruf erfolgt als Ereignis-Handler der Optionsfelder auf der Grafik-Dialogseite.

**Kodierung 17-17: RgRadioButton_Click für die Anwendung 1-Dialogform**

## 17.7.3 GrafikDialogseite_CheckBox_Click

GrafikDialogseite_CheckBox_Click bearbeitet eine Betätigung des Auswahlfeldes der Grafik-Dialogseite für den Umgebungslichtsensor.

Für MULTITHREAD wird vor der Veränderung der globalen Variablen der Bedienerthread mit dem Arbeitsthread synchronisiert.

Der Aufruf erfolgt durch den in InitializeComponent eingetragenen Handler für die Click-Ereignisse der Checkboxen.

**Kodierung 17-18: GrafikDialogseite_CheckBox_Click**

## 17.7.4 Umgebungslicht_Textbox_LostFocus

Umgebungslicht_TextBox_LostFocus verarbeitet die Eingabe in die Textkontrolle für das Umgebungslicht.

Ein korrekt umgewandelter Eingabetext wird zum Umgebungslichtwert. Hierzu wird auch der Wert der ScrollBar angepasst.

Die Raumgrafik für den neuen Umgebungslichtwert wird durch einen Aufruf von EVENT_PROCS.Groessen realisiert.

Der Aufruf erfolgt durch den in InitializeComponent eingetragenen Handler für das LostFocus-Ereignis, wenn der Eingabefocus die Textkontrolle verlässt.

**Kodierung 17-19: Umgebungslicht_TextBox_LostFocus für die Grafik-Dialogseite**

## 17.7.5 Umgebungslicht_ScrollBar_ValueChanged

Umgebungslicht_ScrollBar_ValueChanged verarbeitet die Bedienung der Drehfelder für die Textkontrolle des Umgebungslichts.

Ein korrekter Bildlauffeldwert wird zum Umgebungslichtwert. Hierzu wird auch der Text in der Textkontrolle angepasst.

Die Raumgrafik für den neuen Umgebungslichtwert wird durch einen Aufruf von EVENT_PROCS.Groessen realisiert.

Der Aufruf erfolgt durch den in InitializeComponent eingetragenen Handler für das ValueChanged-Ereignis, wenn der Benutzer das Bildlauffeld verschiebt.

**Kodierung 17-20: Umgebungslicht_ScrollBar_Valuechanged für die Grafik-Dialogseite**

# 18 Die Managed DirectX-Bibliothek

Für Anwendungen werden Managed DirectX Bibliotheken für Direct3D, DirectSound und DirectDraw in der Datei Managed_DirectX.cs zur Verfügung gestellt.

Jede Bibliothek kann durch Ableitung von der Basisklasse oder als separat erzeugte Instanz in die Anwendung integriert werden.

## 18.1 Die Bibliothek in das Projekt einbinden

In Visual Studio müssen zur Verwendung der Methoden von Managed DirectX folgende weitergehende Einstelllungen vorgenommen werden:

1. Unter Verweise müssen die im Windows-Verzeichnis unter Microsoft.Net\DirectX for Managed Code\1.0.2902.0 befindlichen und benötigten dll-Dateien mit dem IL-Code für DirectX hinzugefügt werden.
2. In den Projekt-Eigenschaften muss unter Erstellen die x86-Zielplattform ausgewählt werden.
3. Wenn die Anwendung als Zielframework .Net Framework 4.0 oder höher verwendet, muss in der zum Projekt gehörenden Datei app.config folgender Eintrag vorhanden sein:
   <startup useLegacyV2RuntimeActivationPolicy="true">
   <supportedRuntime version="v4.0"/></startup>.
4. Wenn Punkt 3 zutrifft ist zum Starten der Anwendung zusätzlich zu den DirectX-Dlls auch die vom Compiler erzeugte Datei Appname.exe.config erforderlich.

### 18.1.1 Änderungen zur Version 2.0

- Die Version 1.0 der DIRECT3D-Klasse stellt Zeichenmethoden mit dem Gerätepuffer, mit Texturen und Sprites zur Verwendung bereit.

  Des Weiteren werden Szenen-Methoden zur Steuerung des Renderablaufs, Surface Kopiermethoden und weitere Hilfsmethoden zur Verfügung gestellt.

  Der DIRECT3D-Konstruktor erzeugt eine Direct3D-Geräteinstanz. Die Geräteparameter werden für die Verwendung von Sprites, Texturen und Surfaces in einer Windows Form eingestellt.

### 18.1.2 Änderungen zur Version 1.1

- Um in CreateBuffer eine Endlosschleife bei einem nicht vorhandenem Sound-Stream zu vermeiden, werden der Backbufferindex und die Laufvariable i außerhalb der if-Abfrage erhöht.

- Um in Create3DBuffer eine catch-Ausnahme bei einem ungültigen Klangpuffer zu vermeiden, wird dieser vor der Erzeugung des 3D-Klangpuffers abgefragt.

- In Set3D_Listener wird die Zuhörerposition ersatzweise in die Raummitte gesetzt und alle Änderungen mit CommitDeferredSettings() gültig gegeben. Hierdurch werden in Blocker und Perpetuum mobile die Zuhörerpositionen in die Raummitte immer korrekt gesetzt.

## 18.2 Methoden für die 3D-Grafik mit Direct3D

Die DIRECT3D-Klasse stellt Methoden zur Einbindung von Direct3D zur Verfügung.

**Hinweis 15 zur ersten Version der DIRECT3D-Klasse**

Die in dieser Version zur Verfügung gestellen Methoden ermöglichen eine Einarbeitung in den komplexen Umfang der Direct3D-Bibliothek und erste Ablauftests.

Wir selbst haben versucht damit Perpetuum mobile auf Direct3D zu erweitern.

Hierzu haben wir die .Net-Bitmaps in Texturen und Surfaces umgewandelt. Die Programmtest verliefen erfolgreich.

Es ist uns aber nicht gelungen, die weiße Maske aus den .Net-Bitmaps mit AlphaBlend oder Stencils beim Kopieren der Sprites, Texturen oder Surfaces zu entfernen.

Wer uns hierzu einen Hinweis gibt, wie man mit Direct3D weiß in Texturen transparent kopiert, dem wären wir hierfür sehr dankbar!

**Kodierung 18-1: Die DIRECT3D-Klasse**

```
#region DIRECT3D-Klasse
#if DIRECT3D
///**************************************************************************
/// Version 1.0 vom 15.03.18
///**************************************************************************
[CLSCompliantAttribute(true)]    // markiert die Klasse als CLS-kompatibel
public class DIRECT3D
{

    #region Datenfelder, Konstruktor und Destruktor #endregion

    #region Szenen-Methoden für das 3D-Gerät #endregion

    #region Sprite-Methoden #endregion

    #region Zeichenmethoden mit dem Gerätepuffer #endregion

    #region Zeichenmethoden mit Texturen #endregion

    #region Surface-Kopiermethoden #endregion

    #region Hilfsmethoden für das 3D_Gerät #endregion

}
#endif
#endregion
```

## 18.2.1 Instanzvariable

**Kodierung 18-2: Felder der DIRECT3D-Klasse**

## 18.2.2 Der Konstruktor

Im Konstruktor wird eine Direct3D-Geräteinstanz erzeugt.

Die Geräteparameter werden für die Verwendung von Sprites, Texturen und Surfaces in einer Windows Form eingestellt.

Wurde alles korrekt erzeugt, wird Direct3D mit Direct3DOK true gültig gegeben.

Datenschnittstelle

**Kodierung 18-3: der DIRECT3D-Konstruktor**

## 18.2.3 Ressourcen im Destruktor freigeben

Wegen Problemen mit dem normalen Destruktor, wird DIRECT3D_DESTRUKTOR im Programmablauf aufgerufen, wenn eine neue Device erzeugt wird und zum Programmende.

**Kodierung 18-4: DIRECT3D_DESTRUKTOR**

## 18.2.4 Scenen-Methoden für das 3D-Gerät

### 18.2.4.1 DeviceBeginScene

DeviceBeginScene startet das Zeichnen mit der 3D-Device

**Kodierung 18-5: DeviceBeginScene**

### 18.2.4.2 DeviceSetupCamera

DeviceSetupCamera initialisiert die Kamerausrichtung der 3D-Device

**Kodierung 18-6: DeviceSetupCamera**

### 18.2.4.3 DeviceEndScene

DeviceEndScene beendet das Zeichnen mit der 3D-Device

**Kodierung 18-7: DeviceEndScene**

## 18.2.5 Sprite-Methoden

### 18.2.5.1 SpriteBegin

SpriteBegin beginnt das Zeichnen von Sprites mit der 3D-Device.

**Kodierung 18-8: SpriteBegin**

### 18.2.5.2 SpriteDraw

SpriteDraw zeichnet die Sprite-Oberflächer von der 3D-Device.

**Kodierung 18-9: SpriteDraw**

### 18.2.5.3 SpriteEnd

SpriteEnd beendet das Zeichnen des Sprites von der 3D-Device.

**Kodierung 18-10: SpriteEnd**

#### 18.2.5.4 SpriteFromDevice

SpriteFromDevice erzeugt von der 3D-Device eine Sprite-Oberfläche.

**Kodierung 18-11: SpriteFromDevice**

### 18.2.6 Zeichenmethoden mit dem Gerätepuffer

#### 18.2.6.1 Buffer_Grafikausgabe

Buffer_Grafikausgabe aktiviert die Methoden für die Ausgabe des Scheitelpunktpuffers.

**Kodierung 18-12: Buffer_Grafikausgabe**

#### 18.2.6.2 DeviceDrawTransformedBuffer

DeviceDrawTransformedBuffer zeichnet den positionierten Devicebuffer.

**Kodierung 18-13: DeviceDrawTransformedBuffer**

#### 18.2.6.3 DeviceDrawBuffer

DeviceDrawBuffer zeichnet den Scheitelpunktpuffer der Device.

**Kodierung 18-14: DeviceDrawBuffer**

### 18.2.7 Zeichenmethoden mit Texturen

#### 18.2.7.1 DeviceDrawTransformedTexture

DeviceDrawTransformedTexture zeichnet eine Textur über den positionierten Gerätepuffer.

**Kodierung 18-15: DeviceDrawTransformedTexture**

#### 18.2.7.2 DeviceDrawTexture

DeviceDrawTexture zeichnet eine Textur über den Devicebuffer.

**Kodierung 18-16: DeviceDrawTexture**

### 18.2.8 Surface-Kopiermethoden

#### 18.2.8.1 DeviceCopieSurface

DeviceCopySurface kopiert eine Quelloberfläche in eine Zieloberfläche.

**Kodierung 18-17: DeviceCopySurface**

### 18.2.8.2 DeviceUpdateSurface

DeviceUpdateSurface kopiert eine Quelloberfläche in eine Zieloberfläche.

**Kodierung 18-18: DeviceUpdateSurface**

## 18.2.9 Hilfsmethoden für das 3D-Gerät

### 18.2.9.1 TextureFromBitmap

TextureFromBitmap erzeugt von einer Bitmap eine Texture.

**Kodierung 18-19: TextureFromBitmap**

### 18.2.9.2 SurfaceFromBitmap

SurfaceFromBitmap erzeugt von einer Bitmap eine Oberfläche.

**Kodierung 18-20: SurfaceFromBitmap**

### 18.2.9.3 CreateSurface

CreateSurface erzeugt eine Oberfläche in der Größe vom Formpanel des Basisprogramms.

**Kodierung 18-21: CreateSurface**

### 18.2.9.4 GetRenderTarget

GetRenderTarget liefert die Zieloberfläche der Grafikausgabe.

**Kodierung 18-22: GetRenderTarget**

### 18.2.9.5 DeviceClear

DeviceClear zeichnet den Ausgabebereich der 3D-Device mit einer Farbe.

**Kodierung 18-23: DeviceClear**

### 18.2.9.6 SurfaceColoFill

SurfaceColorFill zeichnet in eine Oberfläche eine Farbe.

**Kodierung 18-24: SurfaceColorFill**

### 18.2.9.7 Schablone

In Schablone werden für den Stencilpuffer die Parameter zum zeichnen einer Maske eingestellt.

**Kodierung 18-25: Schablone**

# 18.3 Methoden für die Klangausgabe mit DirectSound

**Kodierung 18-26: Die DIRECTSOUND-Klasse**

```
#region DIRECSOUND-Klasse
///*************************************************************************
/// Version 1.1 vom 16.12.16
///*************************************************************************
```

```
[CLSCompliantAttribute(true)]
public class DIRECTSOUND
{

    #region Datenfelder, Konstruktor und Destruktor #endregion

    #region CreateBuffer, Create3DBuffer und Set3DBuffer  #endregion

    #region Direct3D_Sound, Set3DListener und Direct3D_Listener #endregion

}
#endregion
```

## 18.3.1 Instanzvariable

**Kodierung 18-27: Felder der DIRECTSOUND-Klasse**

## 18.3.2 eine Instanz mit dem Konstruktor erzeugen

Im Konstruktor wird ein DirectSound-Gerät geöffnet und nach der Prioritätszuweisung hierfür ein Primärer-Buffer erzeugt und ständig wiederholend abgespielt. Von dem Primären Buffer wird ein 3D Listener-Interface abgeleitet. Wurde alles korrekt erzeugt, wird das DirectSound-Gerät für den weiteren Ablauf mit DirectSoundOK true gültig gegeben.

Datenschnittstelle  Struktogramm

**Kodierung 18-28: der DIRECTSOUND-Konstruktor**

## 18.3.3 Ressourcen im Destruktor freigeben

Der DIRECTSOUND-Destruktor gibt Buffer- und Listener-Ressourcen und die DirectSound-Device über Dispose frei.

**Kodierung 18-29: der DIRECTSOUND-Destruktor**

## 18.3.4 Klangpuffer erzeugen

Datenschnittstelle  Struktogramm

In CreateBuffer werden von als Streams eingelesenen Klangdateien die Klangpuffer und Hintergrundpuffer erzeugt und mit den Klangdaten beschrieben. Der Rückgabewert ist true, wenn zumindest alle Klangpuffer erzeugt und beschrieben werden konnten.

**Kodierung 18-30: CreateBuffer**

## 18.3.5 3D Klangpuffer erzeugen

Datenschnittstelle  Struktogramm

In Create3DBuffer werden 3D-Klangpuffer von vorhandenen Klangpuffern ableitend erzeugt.

**Kodierung 18-31: Create3DBuffer**

## 18.3.6 Die Parameter für alle 3D Buffer festlegen

Datenschnittstelle

Set3DBuffer setzt die Parameter aller 3D-Klangpuffer auf grundlegende Werte.

**Kodierung 18-32: Set3DBuffer**

## 18.3.7 Geschwindigkeit und Position für einen 3D Buffer festlegen

Datenschnittstelle

In Direct3D_Sound werden die Position und Geschwindigkeit eines 3D-Klangspeichers auf die Werte eines Objektes gesetzt.

**Kodierung 18-33: Direct3D_Sound**

## 18.3.8 Die Zuhörerschnittellle festlegen

Datenschnittstelle

Set3DListener setzt die 3D Zuhörerschnittstelle auf grundlegende Werte. Zusätzlich wird für Positionsmodus = 1 die Zuhörerposition auf den Bildschirm gesetzt sonst wird der Zuhörer in die Raummitte versetzt.

**Kodierung 18-34: Set3DListener**

## 18.3.9 Geschwindigkeit und Position für den Zuhörer bestimmen

Datenschnittstelle

Für Modus = 3 wird in Direct3D_Listener der Zuhörer auf einen sich im Raum hin- und her bewegenden Fahrstuhl gesetzt. Ist Modus = 4 bewegt sich der Zuhörer mit einem Objekt.

**Kodierung 18-35: Direct3D_Listener**

# 18.4 Methoden für eine synchronisierte Grafikausgabe mit DirectDraw

**Kodierung 18-36: Die DIRECTDRAW-Klasse**

```
#region DIRECTDRAW-Klasse
#if DIRECTDRAW
///*************************************************************************
/// Version 1.0 vom 16.04.14
///*************************************************************************
[CLSCompliantAttribute(true)]
public class DIRECTDRAW
{

    #region Datenfelder, Konstruktor und Destruktor #endregion

    #region SynchronAblauftest und  WfVertRetrace #endregion
```

```
}
#endif
#endregion
```

## 18.4.1 Instanzvariable

**Kodierung 18-37: Felder der DIRECTDRAW-Klasse**

## 18.4.2 eine Instanz mit dem Konstruktor erzeugen

Im Konstruktor wird ein DirectDraw-Gerät geöffnet und eine normale Priorität zugewiesen.

Wurde beides korrekt erzeugt, wird DirectDraw für den weiteren Ablauf mit DirectDrawOK true gültig gegeben.

Datenschnittstelle

**Kodierung 18-38: der DIRECTDRAW-Konstruktor**

## 18.4.3 Ressourcen im Destruktor freigeben

Der DIRECTDRAW-Destruktor gibt die Device über Dispose frei.

**Kodierung 18-39: der DIRECTDRAW-Destruktor**

## 18.4.4 SynchronAblauftest

SynchronAblauftest überprüft, ob der Bildschirm für eine synchronisierte Grafikausgabe mit WfVertRetrace geeignet ist.

**Kodierung 18-40: SynchronAblauftest**

## 18.4.5 den vertikalen Strahlrücklauf abwarten

WfVertRetrace wartet auf das Ende der Grafikausgabe zum Bildschirm (bei Röhrenbildschirmen auf den Beginn des vertikalen Strahlrücklaufs).

Dieses Ereignis liegt bei modernen Monitoren zwischen 60 bis 120 Hz.

Die effektive Nutzung der Prozessorzeit durch das aufrufende Programm kann sich durch den Aufruf dieser Prozedur erheblich verringern.

**Kodierung 18-41: WfVertRetrace**

# 19 Das Multimedia Theater-Projekt

Das Multimedia Theater-Projekt erstellt ein Programm zur Verwaltung von vielfach gestarteten Instanzen der Anwendung 1 und des Basisprogramms selbst. Für die Konfigurationen werden aber nur die Instanzen der Anwendung 1 berücksichtigt.

## 19.1 Die Klassen

Das Multimedia Theater-Projekt besteht aus den Klassen MDI_FORM, MDIFORM_GLOBALS, MDIFORM_TEXTE und MDIFORM_MENUE_TEXTE. Alle Klassen werden in dem Namensraum MDIPrj aufgeführt.

## 19.2 Die MDI_FORM-Klasse

Die Verwaltungsklasse für Anwendungsformen wird direkt von der Form-Klasse abgeleitet.

**Kodierung 19-1: Die MDI_Form-Klasse**

```
#region MDI_FORM-Klasse
///*************************************************************************
/// Version 4.0 vom 15.05.18
///*************************************************************************
[CLSCompliantAttribute(true)]
public class MDI_FORM : Form
{
   #region Main() für das Multimedia Theater-Projekt , globale Ausnahmemethoden und
            InitializeComponent #endregion

   #region Datenfelder, Konstruktor und Kontrollmethode #endregion

   #region StatusBar und Hauptmenü erzeugen #endregion

   #region Bearbeitung der Ereignisse für die MDI-Form #endregion

   #region Bearbeitung der Menü-Ereignisse #endregion

}
#endregion
```

### 19.2.1 Der Haupteinstiegspunkt

Die MDI_FORM-Klasse enthält neben der statischen Methode Main, die durch den Compiler als Startobjekt für das Formenverwaltungsprogramm aufgerufen wird, noch zwei Methoden als Handler von globalen Ausnahmen.

Für einen kontrollierten Ablauf wird in der statischen Main-Methode in einem try-catch-Block ein MDI_FORM-Objekt mit einer separaten Anweisung erzeugt und die Standardmeldungsschleife für das Formenverwaltungsprogramm in einer zweiten Anweisung mit Application.Run gestartet, wodurch auch die MDI-Form sichtbar und bedienbar wird. Nach einer hierin abgefangenen und protokollierten Ausnahme wird das Formenverwaltungsprogramm beendet.

Zum globalen Abfangen und Protokollieren von unbehandelten Ausnahmen werden deshalb zwei Ausnahmehandler erzeugt, nach deren Aktivierung das Formenverwaltungsprogramm fortgeführt werden kann.

**Kodierung 19-2: Die Startmethode Main()**

### 19.2.2 Die Designermethode InitializeComponent

InitializeComponent ist eine für die Designerunterstützung erforderliche Methode. Der Inhalt der Methode darf nicht mit dem Code-Editor geändert werden, weil die MDI-Form soweit erforderlich mit dem Designer verwaltet wird. Die Menüs werden aber manuell kodiert erstellt.

Der Aufruf erfolgt im Konstruktor.

**Kodierung 19-3: InitializeComponent für MDI_FORM**

### 19.2.3 Die Instanzvariablen

Auf alle implizit privaten Instanzvariablen kann nur innerhalb der MDI_FORM-Klasse zugegriffen werden.

**Kodierung 19-4: Die Instanzvariablen der MDI_FORM-Klasse**

### 19.2.4 Der Konstruktor

Der Konstruktor für die MDI_FORM-Klasse erzeugt ein Basisobjekt von MDIFORM_GLOBALS und ruft zum Entwurf der Form mit Designerunterstützung InitializeComponent auf.

**Kodierung 19-5: Der MDI_FORM-Konstruktor**

## 19.3 Die öffentlichen Datenfelder

Datenfelder, die für andere Klassen öffentlich zugänglich sein müssen, werden in einer eigenen Klasse MDIFORM_GLOBALS aufgeführt. Auf diese Datenfelder wird hierdurch über eine separate Referenz zugegriffen. MDIFORM_GLOBALS wird einmal bei der Erzeugung der Instanz im Konstruktor der MDI_FORM-Klasse aufgerufen.

Wenn erforderlich, werden im Konstruktor von MDIFORM_GLOBALS die öffentlichen Datenfelder initialisiert.

Von MDIFORM_GLOBALS werden automatisch Objekte von MDIFORM_TEXTE und MDIFORM_MENUE_TEXTE als Basisklassen abgeleitet. Diese enthalten zwei- oder dreidimensionale statische Arrays, in denen die mehrsprachigen Texte für das Programm und das Menü eingetragen werden.

Die Menütexte enthalten zusätzlich einen kleinen Erläuterungstext zur Menüauswahl, der für die Ausgabe zur Statusleiste verwendet wird.

**Kodierung 19-6: die MDIFORM_GLOBALS-Klasse**

### 19.3.1 Mehrsprachige Ausgabetexte

Die MDIFORM_TEXTE-Klasse enthält mehrdimensionale, öffentlich zugängliche Textarrays für multilinguale Projekt- und Messagebox-Texte der Formenverwaltung und erbt die Menütexte von MDIFORM_MENUE_TEXTE. Eine Instanz wird mit der MDIFORM_GLOBALS-Klasse erzeugt.

Im Konstruktor werden die multilingualen Texte für die Formenverwaltung in den Textarrays bereitgestellt.

Zur Seiteneinsparung werden nur die deutschen und englischen Texte aufgeführt.

**Kodierung 19-7: die MDIFORM_TEXTE-Klasse**

### 19.3.2 Multilinguale Menütexte

Die MDIFORM_MENUE_TEXTE-Klasse enthält mehrdimensionale, öffentlich zugängliche Textarrays für die multilingualen Menütexte der Formenverwaltung mit Erläuterungstexten. Eine Instanz wird mit der MDIFORM_TEXTE-Klasse erzeugt.

Im Konstruktor werden die multilingualen Menütexte für die Anwendung 1 in den Textarrays bereitgestellt.

Zur Seiteneinsparung werden nur die deutschen und englischen Texte aufgeführt.

**Kodierung 19-8: Die MDIFORM_MENUE_TEXTE-Klasse**

## 19.4 Das Hauptmenü

ErzeugeMDIFormMenue erzeugt das multilinguale Hauptmenü für die MDI-Form.

Die Aufrufe erfolgen in den Ereignissen OnLoad, SpracheMenueOnClick und HilfeTextSpracheMenueOnClick sowie in der Kontrollmethode und in MDI_Konfiguration.

Wenn ErzeugeMDIFormMenue nicht von ihrem Erzeuger-Thread aufgerufen wird, wird sie durch den Aufruf eines Delegaten im Erzeuger-Thread erneut aufgerufen.

Die einzelnen Menüelemente werden über die Einträge in den dreidimensionalen Menütext-Arrays zusammengestellt. Hierbei bestimmt die Größe der zweiten Array-Dimension die Anzahl der Menüelemente. Die erste Dimension wählt die Textsprache und die dritte Dimension den Text selbst aus.

Für alle Menüelemente werden Handler für die Select-Ereignisse und, wenn erforderlich, für die Click- und Popup-Ereignisse eingetragen. Der Taste F1 wird als Shortcut der Menüpunkt zur Einführung des Hilfetextes zugewiesen.

Der Titeltext der MDI-Form wird korrigiert, was nach einem Wechsel der Bedienersprache erforderlich ist. Danach wird in StatusBarText_Init der Panel- und Tooltiptext der Statusbar multilingual korrigiert.

**Kodierung 19-9: ErzeugeMDIFormMenue für die MDI-Form**

## 19.5 Die Statusleiste

ErzeugeMDIFormStatusBar erzeugt eine StatusBar-Kontrolle mit einem Panel.

Dem Panel wird ein Text und übergeordnet ein ToolTipText zugewiesen.

Die StatusBar wird nur angezeigt, wenn StBarAnzeige true ist.

Der Aufruf erfolgt in der überschriebenen Methode für die MDI-Form OnLoad.

**Kodierung 19-10: ErzeugeMDIFormStatusBar für die MDI-Form**

## 19.6 Erweiterungen und Änderungen zur Version 4.0

- Für die automatische Größenänderung von Steuerelementen für HighDpi-Anzeigen wird in der Projektdatei App.config der Zusatz <appSettings><add key="EnableWindowsFormsHighDpiAutoResizing" value="true" /></appSettings> Zugefügt.

### 19.6.1 Codeanalyse mit Visual Studio Community

Von der Codeanalyse für den ausgeführten Regelsatz „Alle Microsoft Regeln" werden viele Warnungen angezeigt, die bisher noch nicht alle beseitigt wurden.

Der ausgeführte Regelsatz kann unter den Projekteigenschaften\Codeanalyse ausgewählt werden. Von dem Regelsatz „Alle Microsoft Regeln" ausgehend, wird über den Öffnen-Taster durch Visual Studio eine neue Seite mit einer Liste von Warnungen angezeigt, die über Checkmarken abgewählt werden können.

Danach steht für Multimedia Theater ein eigener Regelsatz zur Auswahl.

Die grundlegend behobenen und abgewählten Regeln werden nur im Basisprojekt aufgelistet.

- Folgende Warnungen wurden behoben:
  CA2210: Assemblys müssen gültige starke Namen aufweisen:
  anhand der Fehlerbeschreibung wurde die Schlüsseldatei MultimediaTheater.snk erstellt.

### 19.6.2 Erweiterungen und Änderungen zur Version 2.0

- Zur weitergehenden Anzeige im Objektkatalog und für die kontextsensitive Hilfe den XML-Kommentar für alle Methoden mit <para>-Tags innerhalb der <summary>- und <remarks>-Tags versehen.
- Neben der Erhöhung der globalen Variablen catch_Ausnahmen, werden die innerhalb des Programmablaufes auftretenden Ausnahmen mit Datum und Uhrzeit in eine Ausnahmeprotokolldatei geschrieben. Hierzu wird im MDI-Verwaltungsprogramm auf OnLoad() ein Ausgabestream für die Datei "MDIProgramm_Exceptions.txt" erzeugt, der auf OnClosed() wieder geschlossen wird (für die als MDI-Formen gestarteten Programme heißt die Ausnahmedatei immer „MDIFormen_Exceptions.txt").
- Zum globalen Abfangen und Protokollieren von unbehandelten Ausnahmen werden in MDI_FORM.Main zwei Ausnahmehandler erzeugt, nach deren Aktivierung die Anwendung fortgeführt wird.
- Das Programm-Menü wird um den Menüpunkt „Konfiguration" erweitert. Das Konfigurationsmenü enthält Auswahlpunkte zum Wiederherstellen, Laden und Speichern des Programmaufbaus als Konfiguration.
- Auf Menüauswahl „Alle Formen Schließen" wird bei mehr als einer geöffneten Form eine Messagebox angezeigt, mit einem Hinweistext zur Abspeicherung des Programmaufbaus als Konfiguration.
- Der MDIFORM_GLOBALS-Klasse wird eine Referenz auf die Dialogform der Anwendung 1 zugefügt. Hierdurch entfallen die Schleifen mit Abfragen aller Kindformen.
- Der AboutBox werden vor der Anzeige über den Menüpunkt „Hilfe/über..." drei LinkLabels zugefügt.
- Im MDI-Projekt werden zur Ermittlung der Programmnummer die Arrays in der Größe der höchsten Programmnummer erzeugt und nicht mehr in der Anzahl der vorhandenen MDI-Kindformen.

- Die Instanzvariable StBarAnzeige der MDIForm_GLOBALS-Klasse wird als Eigenschaft definiert, die zusätzlich die Checkmarke des Hauptmenüs mit verändert.
- In ProgrammMenueOnClick für das MDI-Projekt entfällt zur Anpassung der Panelgröße die Vergrößerung um ein Pixel, da es hierfür unter den neuen .Net Plattformen keinen Grund mehr gibt.

# 20 Die Ereignisse für Multimedia Theater

## 20.1 Ereignisse von der MDI-Form

### 20.1.1 OnLoad

In OnLoad werden die Statusleiste und das Hauptmenü für die MDI-Form und die Instanz zum Schreiben der Fehlermeldungen in eine Datei erzeugt.

Des Weiteren werden die Startdaten aus der Init.-Datei und das Icon aus den Assembly-Ressourcen gelesen und ein Zeitgeber zur Aktivierung der Kontrollmethode erzeugt.

OnLoad überschreibt die geerbte Form-Methode und wird nur einmal aufgerufen bevor die Form angezeigt wird.

**Kodierung 20-1: OnLoad für die MDI-Form**

### 20.1.2 OnMDIChildActivate

Wenn die aktive Kindform eine Anwendung mit einer geöffneten Dialogform ist, werden die Referenzen der Dialogform auf die öffentlichen Datenobjekte der zuletzt aktiven Anwendung gegen die Referenzen der jetzt aktiven Anwendung ausgetauscht. Die aktive Anwendung wird auch zum neuen Eigner der Dialogform. Durch Aufruf von LoadHandler wird danach die Dialogform neu aufgebaut.

OnMdiChildActivate überschreibt die geerbte Form-Methode und wird bei jeder Aktivierung einer Kindform aufgerufen.

**Achtung!** OnMdiChildActivate wird bei der Erzeugung einer neuen Form auf Form.Show vor dem Load-Ereignis für die Form aufgerufen. Deshalb werden die Dialogformen wegen fehlender Initialisierung der Programmdaten nicht aufgebaut, wenn der Titeltext noch nicht gesetzt wurde.

**Kodierung 20-2: OnMdiChildActivate für die MDI-Form**

### 20.1.3 OnClosed

OnClosed schreibt in Programm_Init wichtige Programmdaten für den nächsten Programmstart in die Datei unter Init_Dateiname.

Wenn während des Programmablaufes innerhalb der try-Blöcke catch-Ausnahmen abgefangen wurden, wird die Anzahl in einer Messagebox angezeigt und auf die Ausnahmeprotokolldatei verwiesen.

Erst danach werden noch vorhandene öffentliche Objekte explizit freigegeben, weil diese noch für die Anzeige der Messagebox benötigt werden.

OnClosed überschreibt die geerbte Form-Methode und wird nur einmal aufgerufen bevor die Form geschlossen wird.

**Kodierung 20-3: OnClosed für die MDI-Form**

### 20.1.4 Dispose

Die von Form verwendeten Ressourcen (mit Ausnahme des Speichers) werden freigegeben.

Dispose überschreibt die geerbte Form-Methode und ruft diese selbst als Basismethode auf. Der Aufruf erfolgt nach OnClosed, nachdem die Form geschlossen wurde.

Dispose wurde als Standardmethode unverändert übernommen.

**Kodierung 20-4: Dispose für die MDI-Form**

## 20.2 Ereignisse von dem Menü der MDI-Form

### 20.2.1 Das abschließende Menüereignis

OnMenueComplete durchläuft zur Aktualisierung der Statusleiste StatusBarText_Init.

Weil das Ereignis für das gemeinsame Menü nicht an die aktive Kindform gesendet wird, muss der Text auch hierfür korrigiert werden.

Für die Endversion wird im Kontrollablauf nochmals die Lizenzdatei kontrolliert. Hierzu wird die Kontrollmethode durch einen Timer ausgeführt.

OnMenuComplete überschreibt die von der Form geerbte Methode und wird aufgerufen, nachdem die Menübehandlung abgeschlossen wurde.

**Kodierung 20-5: OnMenueComplete für Multimedia Theater**

### 20.2.2 Ereignisse für das Hauptmenüelement

HauptMenueOnSelect gibt den Hilfetext für den ausgewählten Hauptmenüpunkt zum ersten Panel der StatusBar aus.

Der Aufruf erfolgt als ein in ErzeugeMDIFormMenue für Select-Ereignisse eingetragener Handler, wenn ein Haupt-Menüpunkt ausgewählt wurde.

**Kodierung 20-6: HauptMenueOnSelect für Multimedia Theater**

### 20.2.3 Ereignisse für das Programmmenüelement

ProgrammMenueOnPopup aktiviert oder deaktiviert einzelne, zur Auswahl stehende Programm-Menüpunkte.

Der Aufruf erfolgt als ein in ErzeugeMDIFormMenue für Popup-Ereignisse eingetragener Handler, bevor der Programm-Menüpunkt ausgeklappt wird.

**Kodierung 20-7: ProgrammMenueOnPopup für Multimedia Theater**

ProgrammMenueOnClick bearbeitet die Clicks der Elemente des Programm-Menüs.

Es werden neue Instanzen von der Anwendung 1 und dem Basisprogramm mit sichtbaren MDI-Forms und einem neuen Titeltext erzeugt.

Durch Aufruf der Menüauswahl "Alle Pause / Start !" wird der Pause-Status der Anwendungen gewechselt. Alle MDI-Formen werden auf "Alle Schließen !" geschlossen und das Formen-verwaltungsprogramm auf "Ende !" beendet.

Der Aufruf erfolgt als ein in ErzeugeFormMenues für Click-Ereignisse eingetragener Handler, wenn ein Programm-Menüpunkt angeklickt wurde.

**Kodierung 20-8: ProgrammMenueOnClick für Multimedia Theater**

ProgrammMenueOnSelect gibt den Hilfetext für den ausgewählten Programm-Menüpunkt zum ersten Panel der StatusBar aus.

Der Aufruf erfolgt als ein in ErzeugeMDIFormMenue für Select-Ereignisse eingetragener Handler, wenn ein Programm-Menüelement ausgewählt wurde.

**Kodierung 20-9: ProgrammMenueOnSelect für Multimedia Theater**

### 20.2.4 Ereignisse für das Konfigurationsmenüelement

In KonfigurationsMenueOnPopup werden für diese Programmvariante mit einer Anwendung einzelne Konfigurations-Menüpunkte unsichtbar gemacht.

Für die Testversion ist der „Speichern"-Menüpunkt nicht auswählbar.

Der Aufruf erfolgt als ein in ErzeugeMDIFormMenue für Popup-Ereignisse eingetragener Handler, bevor der Konfigurations-Menüpunkt ausgeklappt wird.

**Kodierung 20-10: KonfigurationsMenueOnPopup für Multimedia Theater**

KonfigurationsMenueOnClick bearbeitet die Clicks der Elemente des Konfigurations-Menüs.

Es werden Programme gestartet oder vorhandene Konfigurationen wieder hergestellt, geladen oder abgespeichert.

Der Aufruf erfolgt als ein in ErzeugeFormMenues für Click-Ereignisse eingetragener Handler, wenn ein Konfigurations-Menüpunkt angeklickt wurde.

**Kodierung 20-11: KonfigurationsMenueOnClick für Multimedia Theater**

KonfigurationsMenueOnSelect gibt den Hilfetext für den ausgewählten Konfigurations-Menüpunkt zum ersten Panel der StatusBar aus.

Der Aufruf erfolgt als ein in ErzeugeMDIFormMenue für Select-Ereignisse eingetragener Handler, wenn ein Konfigurations-Menüelement ausgewählt wurde.

**Kodierung 20-12: KonfigurationsMenueOnSelect für Multimedia Theater**

### 20.2.5 Ereignisse für das Formenmenüelement

FormenMenueOnPopup aktiviert oder deaktiviert einzelne, zur Auswahl stehende Formen-Menüpunkte.

Der Aufruf erfolgt als ein in ErzeugeMDIFormMenue für Popup-Ereignisse eingetragener Handler, bevor der Formen-Menüpunkt ausgeklappt wird.

**Kodierung 20-13: FormenMenueOnPopup für Multimedia Theater**

FormenMenueOnClick bearbeitet die Clicks der Elemente des Formen-Menüs. Der Aufruf erfolgt als ein in ErzeugeFormMenues für Click-Ereignisse eingetragener Handler, wenn ein Formen-Menüpunkt angeklickt wurde.

Die MDI-Formen werden nach Auswahl von "Überlappend", "Nebeneinander" und "Übereinander" neu angeordnet. "Statusleiste" zeigt oder verdeckt die Statusleiste mit gewechseltem Status der Variable StBarAnzeige. Auf "Schließen" wird die aktive MDI-Form und auf "Alle Schließen !" werden alle MDI-Formen geschlossen.

**Kodierung 20-14: FormenMenueOnClick für Multimedia Theater**

FormenMenueOnSelect gibt den Hilfetext für den ausgewählten Formen-Menüpunkt zum ersten Panel der StatusBar aus.

Der Aufruf erfolgt als ein in ErzeugeMDIFormMenue für Select-Ereignisse eingetragener Handler, wenn ein Programm-Menüelement ausgewählt wurde.

**Kodierung 20-15: FormenMenueOnSelect für Multimedia Theater**

## 20.2.6 Ereignisse für das Hilfemenüelement

HilfeMenueOnClick bearbeitet die Clicks der Elemente des Hilfe-Menüs.

Auf "Einführung" und "Menübefehle" wird aus den multilingualen HTML-Hilfetextdateien "Multimedia_Theater4_0_?Language?.chm" ein Inhaltsverzeichnis mit dem Einführungstext oder eine Beschreibung der Menübefehle aufgerufen.

Nach Auswahl von "über..." wird eine ABOUTBOX-Form mit multilingualen Informationen über das Formenverwaltungsprogramm und drei Webseiten-Links angezeigt.

Der Aufruf erfolgt als ein in ErzeugeFormMenues für Click-Ereignisse eingetragener Handler, wenn ein Hilfe-Menüpunkt  angeklickt wurde.

**Kodierung 20-16: HilfeMenueOnClick für Multimedia Theater**

### 20.2.6.1 Das Click-Ereignis für die LinkLabels

In WebLinkLabel_LinkClicked wird nach einem Click auf die LinkLabels innerhalb des Beschreibungstextes der AboutBox ein Systemprozess gestartet.

Der Aufruf erfolgt als ein in HilfeMenueOnClick für Click-Ereignisse von den LinkLabels eingetragener Handler.

**Kodierung 20-17: WebLinkLabel_LinkClicked**

### 20.2.6.2 HilfeMenueOnSelect

HilfeMenueOnSelect gibt den Hilfetext für den ausgewählten Hilfe-Menüpunkt zum ersten Panel der StatusBar aus.

Der Aufruf erfolgt als ein in ErzeugeMDIFormMenue für Select-Ereignisse eingetragener Handler, wenn ein Hilfe-Menüelement ausgewählt wurde.

**Kodierung 20-18: HilfeMenueOnSelect für Multimedia Theater**

## 20.2.7 Ereignisse für das Sprachemenüelement

SpracheMenueOnClick bearbeitet die Clicks der Elemente des Sprache-Menüs.

Auf Click eines Sprachemenüpunktes werden die öffentlichen Daten User_Language und Help_Language verändert und das Menü durch Aufruf von ErzeugeMDIFormMenue in der neuen Sprache erzeugt. Der Paneltext der StatusBar und der Formtitel werden hierbei in der neuen Sprache aktualisiert.

Der Aufruf erfolgt als ein in ErzeugeFormMenues für Click-Ereignisse eingetragener Handler, wenn ein Sprache-Menüpunkt angeklickt wurde.

**Kodierung 20-19: SpracheMenueOnClick für Multimedia Theater**

SpracheMenueOnSelect gibt den Hilfetext für den ausgewählten Sprachemenüpunkt zum ersten Panel der StatusBar aus.

Der Aufruf erfolgt als ein in ErzeugeMDIFormMenue für Select-Ereignisse eingetragener Handler, wenn ein Sprache-Menüelement ausgewählt wurde.

**Kodierung 20-20: SpracheMenueOnSelect für Multimedia Theater**

## 20.2.8 Ereignisse für das Hilfetextsprachemenüelement

HilfeTextSpracheMenueOnClick bearbeitet die Clicks der Elemente des Hilfetextsprache-Menüs.

Auf Click eines Hilfetextsprachemenüpunktes wird das öffentliche Datum Help_Language verändert und das Menü durch Aufruf von ErzeugeMDIFormMenue zur Korrektur der Checkmarke neu erzeugt. Hierdurch kann der Menüpunkt durch die Anwendung verschoben werden.

Der Aufruf erfolgt als ein in ErzeugeFormMenues für Click-Ereignisse eingetragener Handler, wenn ein Hilfetextsprache-Menüpunkt angeklickt wurde.

**Kodierung 20-21: HilfeTextSpracheMenueOnClick für Multimedia Theater**

HilfeTextSpracheMenueOnSelect gibt den Hilfetext für den ausgewählten Hilfetextsprache-Menüpunkt zum ersten Panel der StatusBar aus.

Der Aufruf erfolgt als ein in ErzeugeMDIFormMenue für Select-Ereignisse eingetragener Handler, wenn ein Hilfetextsprache-Menüelement ausgewählt wurde.

**Kodierung 20-22: HilfeTextSpracheMenueOnSelect für Multimedia Theater**

# 21 Der Ablauf von Multimedia Theater

Nach dem Programmstart durch die statische Main-Methode, werden in der von der Form geerbten und überschriebenen Methode OnLoad die Startdaten aus der Initialisierungsdatei geholt und ein Zeitgeber mit einem Intervall von 1 ms für den Aufruf der Kontrollmethode gestartet.

Der Ablauf von Multimedia Theater wird insgesamt in dem für die Anwendung gestarteten Bedienerthread (UI-Thread) durchgeführt.

## 21.1 Die Kontrollmethode

Kontrollmethode stoppt den aufrufenden Timer und führt, wenn erforderlich, zum Programmstart den Kontrollablauf durch. Wenn dieser korrekt abläuft, wird die öffentliche Variable TESTVERSION false.

Ist bei einem Kontrollablauf der Aktivierungsschlüssel nicht korrekt, wird der Anwender über eine Messagebox darüber informiert. Danach werden von allen Programmen die Testversionen gestartet. Ist bei einem Passwort-Kontrollablauf das Passwort nicht korrekt, wird der Programmablauf beendet.

Kontrollmethode wird durch einen in OnLoad erzeugten Timer aufgerufen und läuft somit im Bedienerthread. Dies ist zur korrekten Anzeige der Dialogformen während des Kontrollablaufes über einer sichtbaren Form unbedingt erforderlich.

**Kodierung 21-1: Kontrollmethode für Multimedia Theater**

## 21.2 Die Initialisierung

In Programm_Init werden relevante öffentliche Datenfelder des MDI-Projekts in eine für das Schreiben neu erzeugten Datei unter FileName geschrieben oder aus einer bestehenden Datei eingelesen.

Der Aufruf zum Lesen der Datenfelder erfolgt nach dem Programmstart in OnLoad. Der Aufruf zum Schreiben erfolgt zum Programmende auf OnClosed.

**Hinweis 16 zur Erweiterung der Init-Daten in neuen Versionen des MDI-Programms!**

Für einen fehlerfreien Zugriff auf neu zugefügte Datenfelder in den **alten** Init-Dateien, wird das Datenende auf die konstante MDIDATENENDE vorsorglich nach oben verschoben. Hierbei ist zu beachten das eine string-Erweiterung nur einen Positionswert aus den alten Init-Dateien beansprucht und der eingelesene „string" somit fehlerhaft ist. Integerwerte erhalten auch erstmalig einen falschen Wert, werden aber mit einer korrekten Positionsanzahl eingelesen.

**Kodierung 21-2: Programm_Init für das Formenverwaltungsprogramm**

## 21.3 Die Vergabe der Programmnummer für die MDI-Form

Programmnummer ermittelt für eine neue MDI-Form die erste unbelegte Programmnummer.

Der Aufruf erfolgt in ProgrammMenueOnClick nach der Erzeugung einer neuen Anwendung.

**Kodierung 21-3: Programmnummer für die MDI-Form**

## 21.4 Die Ausnahmeprotokollierung

Ausnahmeprotokollierung erhöht das globale Datenelement catch_Ausnahmen um eins und schreibt für die ersten 25 Ausnahmen den Ausnahmetext mit Datum und Zeit in die Datei der StreamWriter-Instanz SWriter "MDIProgramm_Exceptions.txt".

**Kodierung 21-4: Ausnahmeprotokollierung für Multimedia Theater**

## 21.5 Der Aufbau der MDI-Konfiguration

In MDI_Konfiguration wird über den Parameter Aktion eine MDI-Konfiguration eingelesen, abgespeichert oder wiederhergestellt. Die Aktionen werden nur für die Anwendung 1 und nicht für das Basisprogramm durchgeführt.

Der Rückgabewert ist true, wenn die Aktion korrekt ausgeführt wurde und false nach einem catch-Ablauf, einem Abbruch der Ordnerauswahl oder einer falschen Ordnerauswahl.

Schnittstelle

**Kodierung 21-5: MDI_Konfiguration für Multimedia Theater**

### 21.5.1 Die MDI-Konfiguration Laden

In MDI_Konfiguration_Laden wird die Konfiguration des MDI-Verwaltungsprogramms aus dem Ordner Konfig_FileName eingelesen.

Danach werden alle alten Init-Dateien im Ordner Init_Daten gelöscht und alle in dem Konfigurationsordner vorhandenen Init-Dateien der Programme in diesen Ordner kopiert.

Über einen simulierten Menü-Klick wird für jede in Init_Daten vorhandene Init-Datei ein neues Programm gestartet.

**Kodierung 21-6: MDI_Konfiguration_Laden**

### 21.5.2 Die MDI-Konfiguration Speichern

In MDI_Konfiguration_Speichern werden die Init-Daten von Multimedia Theater und die aller Programme in den Ordner von Konfig_FileName abgespeichert.

Zuvor werden alle alten Init-Dateien der Programme im Konfigurationsordner gelöscht.

**Kodierung 21-7: MDI_Konfiguration_Speichern**

### 21.5.3 Die MDI-Konfiguration Wiederherstellen

In MDI_Konfiguration_Wiederherstellen werden die Init-Daten von Multimedia Theater aus Init_Dateiname wieder eingelesen.

Über einen simulierten Menü-Klick wird für jede im Ordner Init_Daten vorhandene Init-Datei ein neues Programm gestartet.

**Kodierung 21-8: MDI_Konfiguration_Wiederherstellen**

## 21.6 Das Schliessen aller Formen

Nach einer bestätigten Kontrollabfrage über eine Messagebox, werden in Alle_Formen_Schliessen alle Programmformen geschlossen.

Der Rückgabewert ist true, wenn alle Formen korrekt geschlossen wurden und false nach Abbruch durch den Anwender oder einem unkorrekten Ablauf.

**Kodierung 21-9: Alle_Formen_Schliessen für das Formenverwaltungsprogramm**

## 21.7 StatusBarText_Init

Bei einer geladenen Konfigurationsdatei wird zum ersten Panel der StatusBar der Name der Konfigurationsdatei mit relativen Pfadangaben sonst der Starttext des MDI-Programms ausgegeben. Zu beiden Texten wird der Tooltip angepasst.

Die Aufrufe erfolgen in OnMenuComplete, nachdem die Menübehandlung abgeschlossen wurde, und nach der Menüerzeugung in ErzeugeMDIFormMenue.

**Kodierung 21-10: StatusBarText_ Init für für das Formenverwaltungsprogramm**

# 22 Der Aktivierungsschlüssel für die Anwendungen

Das CryptoKeyGenerator-Projekt erstellt eine Form als „FixedToolWindow" mit einer unveränderlichen Größe.

Das Programm erzeugt den für die Anwendungen erforderlichen kryptographischen Aktivierungsschlüssel in einer Binärdatei.

## 22.1 Die CryptoKeyGenerator-Klasse

Der Aktivierungsschlüsselgenerator erzeugt aus einem Registrierungsschlüssel und einem Passwort einen kryptographischen Aktivierungsschlüssel in der Datei "jk-ware_Lizenzdatei.bin".

Zur Kontrolle wird der regenerierte Aktivierungsschlüssel als Text in ein Label der Programmform ausgegeben.

**Kodierung 22-1: die CryptoKeyGenerator-Klasse**

```
#region CryptoKeyGenerator-Klasse
///*********************************************************************
/// Version 2.0 vom 11.10.09
///*********************************************************************
public class CryptoKeyGenerator : Form
{

    #region Instanzvariable, Main() und Konstruktor #endregion

    #region InitializeComponent() und Ereignis-Handler für die Kontrollen #endregion

    #region Enkrypten und Dekrypten des Aktivierungsschlüssels #endregion

}
#endregion
```

### 22.1.1 Die Instanzvariablen

Auf alle implizit privaten Instanzvariablen kann nur innerhalb der CryptoKeyGenerator-Klasse zugegriffen werden.

**Kodierung 22-2: Die Instanzvariablen**

### 22.1.2 Die Startmethode Main()

Main als Haupteinstiegspunkt für die CryptoKeyGenerator-Klasse.

**Kodierung 22-3: Die Startmethode Main()**

### 22.1.3 Der Konstruktor

Der Konstruktor ruft die für die Designerunterstützung der Form erforderliche Methode InitializeComponent auf.

**Kodierung 22-4: Der Konstruktor**

### 22.1.4 Die Designermethode InitializeComponent

InitializeComponent erstellt eine Form als „FixedToolWindow" mit einer unveränderlichen Größe.

Der Inhalt der Methode darf nicht mit dem Code-Editor geändert werden. Der Aufruf erfolgt im Konstruktor der CryptoKeyGenerator-Klasse.

**Kodierung 22-5: InitializeComponent für die CryptoKeyGenerator-Klasse**

## 22.2 Ereignis-Handler für die Formkontrollen des Aktivierungsschlüsselgenerators

### 22.2.1 OnClick_RegkeyEinfuegenButton

OnClick_RegkeyEinfuegenButton kopiert den Text aus der Zwischenablage in die Textbox für den Registierungsschlüssel.

Der Aufruf erfolgt als Handler für das Click-Ereignis des Einfügen-Buttons für den Registierungsschlüssel.

**Kodierung 22-6: OnClick_RegkeyEinfuegenButton des Aktivierungsschlüsselgenerators**

### 22.2.2 OnClick_AnwPwEinfuegenButton

OnClick_AnwPwEinfuegenButton kopiert den Text aus der Zwischenablage in die Textbox für das Anwendungspasswort.

Der Aufruf erfolgt als Handler für das Click-Ereignis des Einfügen-Buttons für das Anwendungspasswort.

**Kodierung 22-7: OnClick_AnwPwEinfuegenButton des Aktivierungsschlüsselgenerators**

### 22.2.3 OnClick_ErzeugenButton

In OnClick_ErzeugenButton wird Erzeuge_Aktivierungsschluessel aufgerufen, worin der Aktivierungsschlüssel erzeugt und in einer Datei abgespeichert wird.

Danach wird der in Dekrypt_Aktivierungsschluessel regenerierte Registrierungsschlüssel zum Label der Form ausgegeben.

OnClick_ErzeugenButton wird als Handler für das Click-Ereignis des Erzeugen-Button aktiviert.

**Kodierung 22-8: OnClick_ErzeugenButton des Aktivierungsschlüsselgenerators**

## 22.3 Enkrypten und Dekrypten des Aktivierungsschlüssels

### 22.3.1 Erzeuge_Aktivierungsschluessel

Erzeuge_Aktivierungsschluessel erzeugt den Aktivierungsschlüssel und schreibt ihn in eine neu erzeugte Datei unter Dateiname.

Der Aufruf erfolgt in OnClick_ErzeugenButton.

**Kodierung 22-9: Erzeuge_Aktivierungsschluessel**

### 22.3.2 Dekrypt_Aktivierungsschluessel

Regeneriert den Aktivierungsschlüssel aus den Dateidaten unter Dateiname.

Der Aufruf erfolgt in OnClick_ErzeugenButton().

**Kodierung 22-10: Dekrypt_Aktivierungsschluessel**

# 23 Der erweiterte Lizenzschutz im LicenseProtectorSample-Projekt

Zur Absicherung der Anwendungen mit dem Lizenzschutz der Mirage GmbH, wird das von dem Unternehmen zur Verfügung gestellte Beispielprojekt übernommen.

Zum korrekten Ablauftest muss allerdings eine neue Lizenzdatei angefordert werden oder besser noch eine neue Version des Licence Protector von www.mirage-systems.de heruntergeladen und installiert werden.

Das Beispielprogramm ruft nach Betätigung des Form-Buttons die Methode protectMe() auf. Darin verursacht der Aufruf der LicProtectorDLL310() wegen fehlender Dll-Library eine Ausnahme.

Die Integration des License Protector in die Projekte wird im Kapitel „Der Mirage-Lizenzschutz" erläutert.

# 24 Übersichtsdiagramm und Struktogramme

- Übersichtsdiagramm

BASISPROJEKT:

- Struktogramme
- Strukturblöcke

ANWENDUNG1:

- Struktogramme

# 25 Kodierungen

# 26 Hinweise