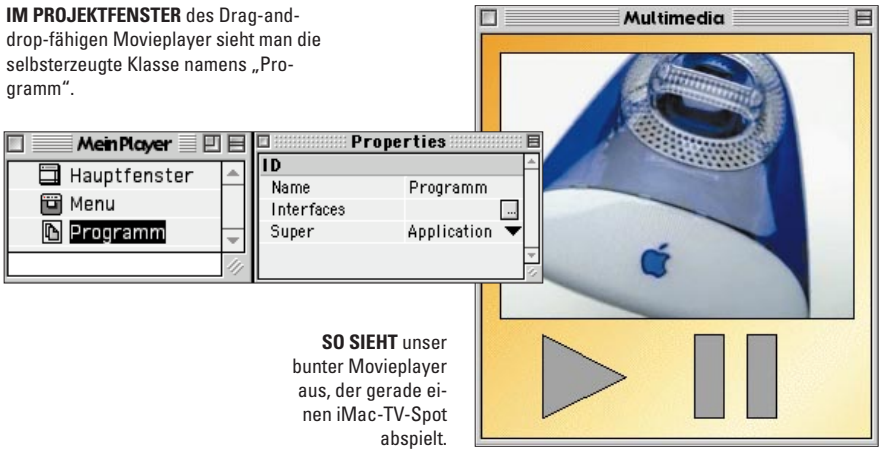
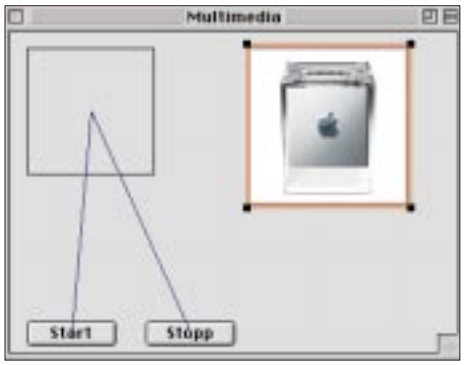


IM PROJEKTFENSTER des Drag-and-drop-fähigen Movieplayer sieht man die selbsterzeugte Klasse namens „Programm".

SO SIEHT unser bunter Movieplayer aus, der gerade einen iMac-TV-Spot abspielt.



BUTTON MIT BILD Buttons müssen nicht grau sein. Man kann in Realbasic auch aus beliebigen Grafiken Knöpfe erzeugen.

ALLES SO SCHÖN BUNT HIER: MOVIEPLAYER IM EIGENBAU

IN DIESEM MONAT wollen wir uns mit Multimedia-Programmierung, also Bildern, Tönen und Filmen beschäftigen. Als Ziel setzen wir uns ein kleines Multimedia-Programm, das man sehr leicht zu einer einfachen Präsentationssoftware oder einer interaktiven Grafikanwendung, wie man sie von vielen Multimedia-CDs her kennt, ausbauen kann

VON CHRISTIAN SCHMITZ

UNSEREN ERSTEN KLEINEN Movieplayer erstellen wir mit lediglich zwei Objektverknüpfungen, ohne eine einzige Programmzeile zu schreiben. Dazu starten wir wie immer mit einem leeren Projekt. Für unser Multimedia-Programm benötigen wir etwas Bild- und Filmmaterial. Dazu suchen wir uns einen Quicktime-Film und ziehen ihn per Drag-and-drop in das Projektfenster. Später brauchen wir noch zwei Pixel-Bilder mit identischen Abmessungen. Noch ein Hinweis: Bitte beachten Sie, dass „Objektverknüpfungen" eine spezielle Funktion von Realbasic 2.x ist und daher nicht mit Realbasic 1.0 beziehungsweise Realbasic Light funktioniert. Für diese Realbasic-Versionen haben wir unsere Beispielprogramme etwas modifiziert. Auf der Heft-CD finden Sie die Quellcodes zu den modifizierten Versionen.

Wie Sie im Screenshot vom Projektfenster weiter vorne auf der Seite 154 sehen, hat Realbasic die Dateinamen zurechtgestutzt. Das muss auch so sein, denn diese Bezeichnungen dienen im Programmcode als Variablennamen, damit man direkt auf die Bilder und Filme im Projekt zugreifen kann. Alle Film- und Sounddateien, die der Quicktime-Player abspielt, lassen sich direkt in das Projekt ziehen und als Movie-Objekt unter dem jeweiligen Namen ansprechen. Es ist außerdem möglich, alle Bilder, die der Quicktime-Picture-Viewer anzeigen kann, ins Projekt zu ziehen und als Picture-Objekte zu verwenden. System-7-Tondateien (zum Beispiel Warntöne oder Effekte) landen direkt als Soundobjekte im Projekt.

QUICKTIME IST VIEL MEHR als nur ein Movieplayer-Alias auf dem Schreibtisch. Apple stellt mit Quicktime eine umfangreiche Funktionenbibliothek zur Verfügung, die jeder Programmierer für seine Zwecke nutzen kann. Auch unter Realbasic ist es möglich, einfach auf den mächtigen Quicktime-Befehlsumfang zurückzugreifen. Wenige Zeilen Programmcode reichen für einen eigenen Movieplayer aus

MIT DEM EIGENEN MOVIEPLAYER HEISST ES: FILM AB!

Da nun alle benötigten Dateien im Projekt als Alias vorliegen, gehen wir zum Hauptfenster über. Aus der Werkzeugpalette ziehen wir das Movieplayer-Steuerelement in unser Hauptfenster. In der Eigenschaftenpalette wählen wir im Pop-up-Menü bei der Eigenschaft „Movie" unseren Film aus. Jetzt können wir das Programm schon starten und den Film abspielen. Einige der Eigenschaften des Movieplayer-Steuerelements sind bereits per Default eingeschaltet. So sorgt die Option „AutoResize" zum Beispiel dafür, dass der Movieplayer genau so groß ist wie der Film. Wenn man diese Option ausschaltet, skaliert Realbasic den Film entsprechend unserem Kontrollelement. Der Film lässt sich also beispielsweise in doppelter Größe abspielen, indem wir die „AutoResize"-Eigenschaft ausschalten und das Kontrollelement im Fenster auf die doppelte Filmgröße aufziehen.

Mit der Eigenschaft „Speaker" erscheint automatisch der Lautstärkeregler unter dem Film, und die Option „HasStep" zeigt die Vor- und Rückspultasten. Diese Funktionen

sind fest in Apples Quicktime eingebaut, und Realbasic setzt komplett darauf auf. Dank an Apple, denn das spart uns eine Menge Programmierarbeit.

INTERFACE-PROGRAMMIERUNG EINFACH PER DRAG-AND-DROP

Ziehen Sie noch zwei Buttons (Push-Buttons oder Bevel-Buttons) in das Fenster. Wenn Sie mit der Maus einen Button selektieren, können Sie direkt losschreiben und die Eigenschaft „Caption“ des Buttons ändern. Auf jeden Fall nennen wir die Buttons „Start“ und „Stop“, denn diese Funktionen sollen Sie später im Programm haben.

Jetzt wollen wir eine Verknüpfung vom Movieplayer zu dem Start-Button erzeugen. Wir drücken die Tasten Befehl-Wahl-Umschalt und klicken auf den Startknopf. Mit gedrückter Maustaste ziehen wir den Mauspfeil auf das Movieplayer-Kontrollelement. Während wir ziehen, folgt eine Linie von der Maus zum Startknopf, und das Movieplayer-Steuerelement lässt sich erfassen (siehe Abbildung Seite 150). Wenn wir die Maustaste loslassen, erscheint der Dialog „New Binding“ und zeigt uns die Verknüpfungen. Wie Sie sehen, kann ein Button, wenn er gedrückt wird, den Film entweder starten oder stoppen. Für den Button „Start“ wählen wir also die Verknüpfung „Play Movie Player1 movie …“ und für den „Stopp“-Button dementsprechend die Verknüpfung „Stop Movieplayer1 movie …“.

Starten wir unser Programm jetzt, können wir den Film mit unseren eigenen Steuerknöpfen abspielen und wieder stoppen.

Dieses Beispiel finden Sie unter dem Namen „Multimedia 1“ auf der Heft-CD. Auch für Realbasic Light gibt es dort eine Version, die jedoch auf die Verknüpfungen verzichten muss und stattdessen mit den Befehlen „movieplayer1.play“ und „movieplayer1.stop“ in den Action-Events der jeweiligen Buttons arbeitet.

BUNT ANIMIERTE BUTTONS MIT DEM ROLLOVER-EFFEKT

Als Nächstes basteln wir zusammen einen eigenen animierten Multimedia-Button, der sich verändert, wenn man mit der Maus darüber fährt. Dazu ziehen wir ein Canvas-Steuerelement in das Fenster und setzen die Eigenschaft „Backdrop“ in der Palette auf das erste Bild im Projekt. Bei einem Teststart des Programms sollte nun schon das erste Bild im Fenster zu sehen sein.

Wieder zurück im Fenstereditor öffnen wir den Code-Editor des Canvas per Doppelklick. Unter den vielen Events, die ein Canvas anbietet, finden wir auch den Event „MouseEnter“. Dort tragen wir als Programmcode die Zeile „me.backdrop=imac blau“ ein, wobei „imacblau“ hier für das zweite Bild steht. Falls Sie ein anderes Bild benutzen wollen, passen Sie den Namen einfach an. Im Event „MouseExit“ kommt parallel dazu der Befehl „me.backdrop=cube“, wobei „cube“ hier der Name des ersten Bildes ist. „me“ bedeutet, dass der Programmcode eine Eigenschaft seines eigenen Objekts anspricht. Man könnte stattdessen den Namen des Objekts benutzen, aber „me“ ist kürzer und eleganter.

Die Eigenschaft „Backdrop“ von einem Canvas oder auch einem ganzen Fenster enthält das Bild, das als Fensterhintergrund angezeigt wird. Fährt man nun zur Laufzeit mit der Maus über das Bild, ruft Realbasic unseren Eventhandler auf, und der Programmcode ändert blitzschnell das Hintergrundbild. Verlässt die Maus den Canvas wieder, passiert das Gleiche, nur dass jetzt das ursprüngliche Bild erscheint.

Um den Redraw der Objekte brauchen wir uns glücklicherweise nicht zu kümmern, denn sobald die Backdrop-Eigenschaft im laufenden Programm geändert wird, führt Realbasic den Redraw des Canvas automatisch durch. Auf der aktuellen Heft-CD finden Sie dieses Beispiel unter dem Namen „Multimedia 2“.

JETZT KOMMT BEWEGUNG INS SPIEL: BILDWECHSEL BEIM MAUSKLICK

Damit der Benutzer beim Mausklick auf den Multimedia-Button auch ein visuelles Feedback bekommt, fügen wir in den Events „MouseDown“ und „MouseUp“ noch etwas Code ein (Beispiel: „Multimedia 3“ auf der Heft-CD). Dafür benötigen wir erst einmal ein weiteres Bild. Also noch schnell ein drittes Bild suchen und per Drag-and-drop auf das Projekt ziehen. In den „MouseDown“-Event des Canvas schreiben wir die Zeile „me.backdrop=ibook“, damit das Bild mit dem Namen „iBook“ angezeigt wird. Im Event „MouseUp“ müssen wir passend dazu ein „me.backdrop=imacblau“ einfügen, um das vorherige Bild wieder zu restaurieren. Wenn man jetzt das Programm startet, klappt das nicht richtig. Genau genommen wird „MouseUp“ nie aufgerufen. Warum es nicht aufgerufen wird, ist eine Standardfrage von Realbasic-Neulingen. Aber schauen wir uns den Event doch einmal an. Da steht „Function MouseDown() as boolean“. Der Event ist eine Funktion, liefert also einen Wert zurück. Wir ergänzen in diesem Event hinter unserer Zeile die Zeile „Return true“. Damit sollte auch das „Mouse Down“ klappen.

Viele Events, zum Beispiel alle Menü-Handler und auch der „Keydown“-Event erwarten, dass man mit „return true“ meldet, ob man den Event verarbeitet hat. Ansonsten wird er eine Instanz höher, also an das Steuerelement, das Fenster oder das ganze Programm, weitergereicht.

PER KNOPFDRUCK ALLES UNTER KONTROLLE: „FILM AB!“

Wir möchten gerne, dass der Film per Mausklick auf unseren eigenen Multimedia-Button startet. Dazu ergänzen wir im „Mouse Up“-Event die Zeile „movieplayer1.play“.

DIE REALBASIC-SPEICHERVERWALTUNG

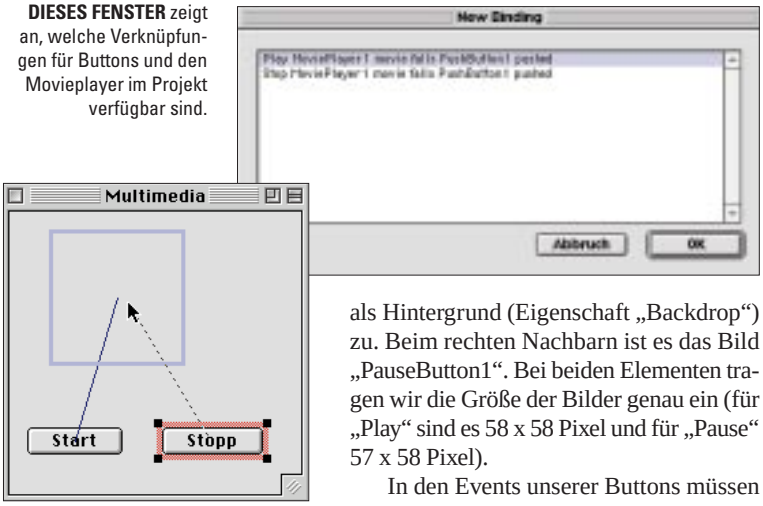
Bei dem vielen Hin- und Herkopieren von Bildern stellt sich sicher die Frage, ob und wann Realbasic mal in Speichernot kommt. Dazu ein paar beruhigende Worte zur internen Speicher- und Variablenverwaltung von Realbasic:
Wenn wir ein Bild laden, reserviert Realbasic intern dafür Speicher im RAM. Jedes Objekt im Speicher verfügt über einen Zähler, der festhält, wie viele Variablen als Referenzen auf das Objekt existieren. Bei dem Befehl „me.backdrop=bild“ wird zuerst einmal der Zähler von dem Bild, auf das „backdrop“ zeigt, um eins verringert. Da jetzt keine Referenz mehr existiert, wird das Bild komplett aus dem Speicher entfernt und anschließend das neue Bild eingeladen. Um die Speicherverwaltung brauchen wir uns als Realbasic-Programmierer also erfreulicherweise nicht zu kümmern, das macht Realbasic von selbst im Hintergrund. Sollte der Speicher dennoch einmal knapp werden, bekommen wir statt des Objekts den Nullwert „nil“ zurück, und der Benutzer sieht einfach nichts. Es empfiehlt sich, im Hinterkopf zu behalten, dass Realbasic alle Bilder im Projekt beim Programmstart direkt einlädt und bis zum Programmende im Speicher hält. Wer Speicher sparen möchte, sollte also möglichst kleine Bilder verwenden oder sie zur Laufzeit aus Bilddateien nachladen.

Damit teilen wir dem Movieplayer-Steuerelement mit, dass es den Film starten soll. Unser Beispiel „Multimedia 4“ auf der Heft-CD macht genau das.

Wir haben nun gelernt, wie man mit drei Pixel-Bildern multimediale Buttons erzeugt. Unser Beispiel „Multimedia 5“ geht noch weiter und stellt einen richtigen Stand-alone-Movieplayer dar.

FARBE MUSS HER: DER BUNTE MOVIEPLAYER

Auf der Heft-CD finden Sie neben dem Programmcode die Grafiken, die für einen bunten Movieplayer sorgen. Konkret sind das jeweils drei Bilder für den Start- und den Pauseknopf, eins für den Knopf im Ruhezustand, eins für den Knopf, wenn die Maus darüber fährt und eins, wenn man die Maustaste drückt. Zusätzlich benötigen wir ein weiteres Bild, das als allgemeiner Fensterhintergrund dient – der Einfachheit halber



DIESES FENSTER zeigt an, welche Verknüpfungen für Buttons und den Movieplayer im Projekt verfügbar sind.

DURCH ZIEHEN mit der Maus stellt man eine Verknüpfung von einem Button zu einem Movieplayer-Objekt her.

erzeugen wir mit Apple Works einen Farbverlauf, jedes andere Bild funktioniert auch.

Wir importieren zunächst alle sieben Grafiken und den Film per Drag-and-drop in das Projektfenster eines neuen Projekts. Damit um die Bilder kein weißer Rand sichtbar bleibt, wählen wir im Projektfenster die Bilder für die Knöpfe einzeln an und setzen jeweils in der Eigenschaftenpalette die Eigenschaft „Transparent“ auf „white“. Damit wird die Farbe Weiß (im RGB-Farbmodell 100% Rot, 100% Grün und 100% Blau) transparent gezeichnet.

Im Hauptfenster platzieren wir wieder ein Movieplayer-Kontrollelement für den Film. Es sollte genau so groß sein wie unser Film (der iMac-Film ist 240 x 180 Pixel groß). Die Eigenschaft „Controller“ des Movieplayer-Kontrollelements setzen wir auf „None“, damit der Benutzer unsere Multimedia-Buttons für die Bedienung nimmt und nicht die Knöpfe von Quicktime. Die Eigenschaft „Autoresize“ können wir wahlweise ein- oder ausschalten. Sie wird nicht mehr gebraucht, stört aber auch nicht. Klicken wir in den leeren Fensterteil, lässt sich in der Palette der Titel des Fensters ändern. Jedes Fenster hat die Eigenschaft „Backdrop“, in der festgehalten wird, welches Bild im Fensterhintergrund erscheint. Dort wählen wir unser Bild „Hintergrund“ aus. Damit bekommt das Fenster einen bunten Hintergrund. Auf Wunsch kann man zudem die Eigenschaft „Frame“ des Fensters ändern. Dort wird angegeben, ob das Fenster als normales Dokumentenfenster („Document Window“) oder beispielsweise als randloses Fenster („Plain Window“) erscheinen soll. Das Fenster muss für unser Beispiel mindestens 266 Pixel breit und 276 Pixel hoch sein.

Wieder im Fenster selbst setzen wir zwei Canvas-Steuerelemente unter den Movieplayer. Sie dienen uns als Buttons. Dem linken Fenster weisen wir in der Eigenschaftenpalette das Bild „PlayButton1“ als Hintergrund (Eigenschaft „Backdrop“) zu. Beim rechten Nachbarn ist es das Bild „PauseButton1“. Bei beiden Elementen tragen wir die Größe der Bilder genau ein (für „Play“ sind es 58 x 58 Pixel und für „Pause“ 57 x 58 Pixel).

In den Events unserer Buttons müssen wir nun noch den Programmcode wie im Beispiel „Multimedia 4“ eintragen. Also schreiben wir jeweils bei „MouseEnter“ „me.backdrop=playbutton2“ und „me.backdrop=pausebutton2“ und im Event „Mouse Exit“ „me.backdrop=playbutton1“ sowie „me.backdrop=pausebutton1“ hinein. In den „MouseDown“-Events laden wir das jeweils dritte Bild per „me.backdrop=playbutton3“ und „me.backdrop=pausebutton3“.

Bei den „MouseUp“-Events schreiben wir „me.backdrop=playbutton2“ und „me.backdrop=pausebutton2“. Vorsicht: hier nicht die Zeile „return true“ vergessen. Durch „movieplayer1.play“ beziehungsweise „movieplayer1.stop“ in den „Mouse Down“-Events der Buttons starten und stoppen wir dann den Film. Nun ist unser Movieplayer bereits einsatzbereit.

DRAG-AND-DROP AUF DAS MOVIEPLAYER-ICON: DAS STECKT DAHINTER

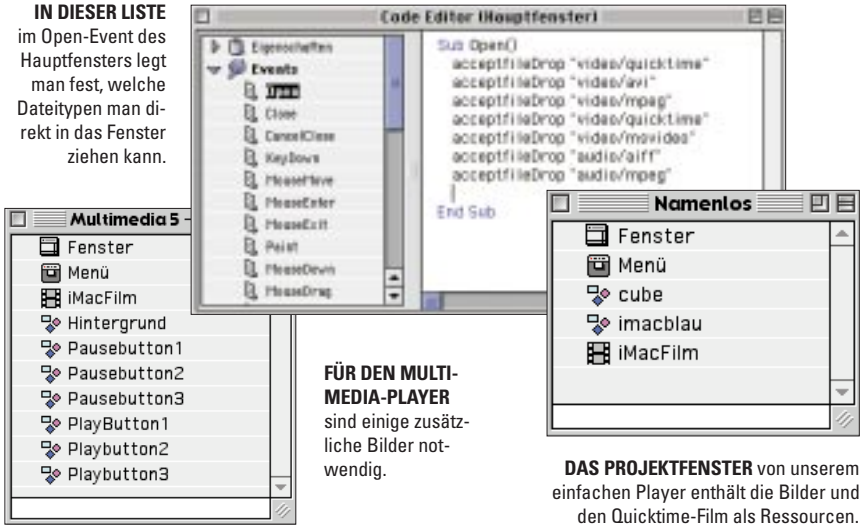
Bislang spielt unser Movieplayer nur einen bestimmten, bereits vorgeladenen Film ab. Viel schöner wäre eine Lösung, die beliebige Filme verarbeitet und sie lädt, wenn man sie per Drag-and-drop im Finder auf den Movieplayer schiebt. Sinnvoll wäre zudem ein Fenster, in dem man die Lautstärke regeln und den Film starten und unterbrechen kann. Dieses Beispiel programmieren wir nun in Realbasic 2 Standard/Pro. Es sollte aber – mit kleinen Änderungen – auch in der Light-Version machbar sein.

Wieder beginnen wir mit einem leeren Projekt und benennen das Standardfenster (Fenster1) in „hauptfenster“ um. Über das Ablagemenü fügen wir eine neue Klasse hinzu. In der Eigenschaftenpalette ändern wir die Eigenschaft „Super“ auf „Application“ und nennen die Klasse „Programm“. Diese Klasse beinhaltet automatisch ein paar Events. Als Erstes schreiben wir in den Event „NewDocument“ die Zeile „msgbox "Kein Dokument aufgerufen"“. Wenn man jetzt das Programm startet, wird ein Warndialog mit dieser Meldung ausgegeben. Nun öffnen wir im Menü „Bearbeiten“ den Dialog „Dateitypen...“. Dort erstellen wir für jeden Dateityp, den unser Programm öffnen können soll, einen Realbasic-Dateityp. Klickt man auf den Button „Neu…“, lassen sich links in einem Pop-up-Menü vorgefertigte Dateitypen auswählen. Wir legen also für jeden Audio- und jeden Videodateityp einen in Realbasic an. Bitte denken Sie an das Häkchen bei „Dokument Icon“, denn ohne es funktioniert Drag-and-drop auf das Programmsymbol nicht.

Wenn wir eine Datei auf das kompilierte Programm im Finder ziehen, schaut der Finder nach, ob unser Programm diesen Dateityp erlaubt. Falls ja, übergibt er die Datei an das Programm, es empfängt eine Meldung des Finders und ruft in der Klasse „Programm“ den Event „OpenDocument“ auf. Als Parameter hat dieser Event ein „DragItem“. Darin enthalten ist unter anderem auch ein „FolderItem“-Objekt, das uns als Alias auf die eigentliche Datei dient. Wir ergänzen also im Event „OpenDocument“ noch eine Zeile „run item“. Den Parameter des Event, nämlich das Folder-Item „item“, das auf die Datei verweist, übergeben wir dem Unterprogramm beziehungsweise der Methode „Run“, die wir später noch programmieren wollen. Vorsicht: Ist in der Dateitypenliste auch nur ein Eintrag mit „????“ für den Typ und „????“ für den Creator vorhanden, akzeptiert das Programm alle Dateien. Das wäre für unseren Player allerdings in Ordnung, denn Dateitypen, die er nicht kennt, ignoriert das Programm einfach.

Nun legen wir noch schnell eine neue Methode namens „Run“ an (Menüpunkt „Neue Methode…“ im „Bearbeiten“-Menü anklicken). Dabei tragen wir als Übergabeparameter „datei as folderitem“ ein. Die Methode soll diese Datei später laden, im Hauptfenster anzeigen und abspielen.

Dazu müssen wir erst einmal das Hauptfenster modifizieren. Wir ändern die Fenstergröße auf einen relativ kleinen Wert von beispielsweise 200 x 100 Pixel. Darüber hinaus schalten wir die Eigenschaften „Grow Icon“ und „ZoomIcon“ in der Palette aus, denn das Fenster ändern wir automatisch vom Programm aus. Im Fenster platzieren wir anschließend ein Movieplayer-Steuerelement in Fenstergröße, jedoch ohne Rand, also left= -1, top=-1, width=202, height= 102. Dann schalten wir die vier Lock-Eigenschaften (Left, Top, Right, Bottom) des Movieplayer-Objekts ein, damit er sich automatisch an die Fenstergröße anpasst. Das „MoviePlayer“-Kontrollelement benennen wir um in „Player“. Ein Doppelklick auf das Fenster (oder den Player) öffnet wie immer den Code-Editor, und wir können eine Methode namens „lade“ anlegen, die als Übergabeparameter die Zeile „datei as folder item“ bekommt.



IN DIESER LISTE im Open-Event des Hauptfensters legt man fest, welche Dateitypen man direkt in das Fenster ziehen kann.

FÜR DEN MULTIMEDIA-PLAYER sind einige zusätzliche Bilder notwendig.

DAS PROJEKTFENSTER von unserem einfachen Player enthält die Bilder und den Quicktime-Film als Ressourcen.

Zurück in der Programmklasse öffnen wir das Unterprogramm „Run“ und rufen dort mit der Programmzeile „hauptfenster.lade datei“ das Unterprogramm des Hauptfensters auf.

Nun muss das Unterprogramm „lade“ noch die Datei öffnen. Dazu benutzen wir die Funktion „openasmovie“. Vorher legen wir mit dem Befehl „dim film as movie“ eine Variable ab, in der wir den Film festhalten. Wir laden den Film aus der Datei mit der Zeile „film=datei.openasmovie“. Doch zu diesem Zeitpunkt wissen wir noch nicht, ob die Datei überhaupt einen Film enthält. Zum Glück nimmt uns auch hier Quicktime die Arbeit ab, denn es meldet uns „nil“ zurück, wenn es sich bei der Datei nicht um ein Quicktime-Dokument handelt. Andernfalls erhalten wir ein Movie-Objekt in der Variable zurück. In der nächsten Zeile brauchen wir also nur zu prüfen, ob wir „nil“ zurückbekommen oder nicht. Dazu benutzen wir: „If film<>nil then“, und weiter unten kommt schon mal ein „end if“ hin, um die Bedienungsabfrage zu schließen.

Zwischen den beiden Zeilen ist jetzt gesichert, dass es sich bei der Datei um einen Film handelt. Wir setzen die Höhe des Fensters mit „player.height=m.movieheight +14“ auf die Filmhöhe plus 14 Pixel für die Quicktime-Steuerleiste (eigentlich müssten es 16 Pixel sein, aber wir arbeiten hier ohne den Rand). Danach ermitteln wir die Breite. Wenn die Datei keinen Videokanal hat (beispielsweise bei einer reinen Audiodatei im MP3-Format), ist die Breite Null. Das ist unangenehm, da nichts mehr ins Fenster passt, wenn dessen Breite kleiner als 50 Pixel wird. Mit dem Vergleich „if film.movie width<50 then“ prüfen wir, ob die Breite zu klein ist und setzen sie danach mit „width=150“ auf 150 Pixel fest. Dabei wächst der Player automatisch auf die passende Größe, denn mit den Eigenschaften „LockLeft“ und „LockRight“ achtet Realbasic selbst automatisch darauf. Nach einem „else“-Befehl setzen wir die Alternative: „width=film.moviewidth“ ändert die Fenstergröße auf die Filmbreite. Nach dem obligatorischen „end if“ müssen wir den Film nur noch per „player.movie= film“ zuweisen. Durch diesen Befehl erscheint der Film im Fenster. Die Videokassette wird sozusagen in den Videorekorder eingelegt, nachdem wir sie aus dem Schrank (Festplatte) geholt haben.

Damit unser Programm funktioniert, müssen wir in den Projekteinstellungen („Bearbeiten“-Menü) noch einen Creator-Code vergeben. Spontan schreiben wir mal „test“ hinein, möchte man das Programm jedoch öffentlich verbreiten, muss man bei Apple einen freien Creator-Code beantragen. Den Code kann man zunächst frei wählen und dann auf der Apple-Internet-Seite http://developer.apple.com/dev/cftype/ überprüfen, ob er noch frei ist. Wenn ja, sollte man ihn schleunigst anmelden, bevor ihn jemand anderer wegschnappt. Aber Achtung, alle Codes, die nur aus Kleinbuchstaben bestehen, sind explizit für Apple reserviert.

Noch sind wir nicht ganz fertig. Fügen wir die Zeile „player.play“ ein, spielt unser Programm den Film direkt nach dem Ladevorgang ohne weiteres Zutun ab.

SO WIRD ES MAC-LIKE: DRAG-AND-DROP INS FENSTER

Eine schöne und Mac-typische Funktion ist Drag-and-drop auf Fenster. Unser Player soll also auch auf Dateien reagieren, die wir direkt in das Movieplayer-Fenster ziehen. Dazu müssen wir im Open-Event des Hauptfensters mit dem Befehl „AcceptFileDrop“ jeden Dateityp, den man ziehen darf, anmelden. Auszugsweise sieht das so aus:

```
AcceptFileDrop "video/quicktime"
AcceptFileDrop "video/avi"
AcceptFileDrop "audio/mpeg"
```

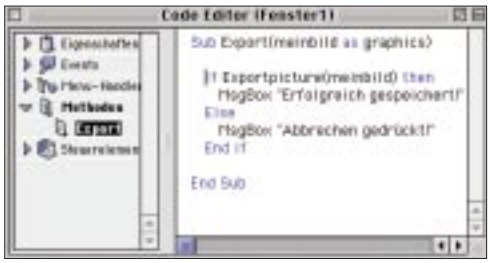
Wichtig ist, dass wir jeden Dateityp mit seinem genauen Namen einzeln nennen.

FILME HERAUSFILTERN

Der Event „DropObject“ bekommt beim Aufrufen einen Parameter „Obj“ vom Typ „DragItem“ übergeben. Dieser enthält das Folder-Item für die Datei. Da wir aber auch andere Dinge als nur Dateien per Drag-and-drop ziehen können, etwa Bilder, Texte oder weitere programmspezifische Daten, müssen wir uns unser Folder-Item selbst heraussuchen. Dazu gehen wir alle Objekte der Reihe nach durch und schauen nach, ob es sich dabei um ein Folder-Item und damit um unsere Datei handelt.

Wir benutzen eine „Do-Loop“-Schleife. Am Anfang steht die Zeile „Do“ als Einleitung einer Do-Loop-Schleife und am Schluss die Zeile „Loop until not Obj.Next Item“. Am Schleifenende rufen wir die Funktion „NextItem“ des Objekts auf. Diese Funktion liefert den Wert „true“ (wahr) zurück, so lange noch ein Objekt übrig ist, und lädt es dann. Gibt es kein weiteres Objekt mehr, bekommen wir den Wert „false“ (falsch) geliefert. Damit unser Programm die Schleife bei „false“ beendet, müssen wir über die logische Verknüpfung „not“ dafür sorgen, dass „true“ und „false“ quasi gegeneinander vertauscht werden (wir merken uns: „not false“ ergibt „true“ und „not true“ ergibt „false“). Und wenn wir jetzt als Ergebnis der Schleife insgesamt ein „true“ erhalten, beendet das Programm die Schleife, und wir können sicher sein, dass wir alle Objekte durchgesehen haben.

Innerhalb der Schleife müssen wir schauen, ob sich im aktuellen Stand des „DragItems“ ein „FolderItem“ befindet. Das fragen wir durch „if obj.folderitemavailable



MIT QUICKTIME lassen sich Pixel-Bilder und Grafiken auch wieder speichern. Die Funktion „ExportPicture" sorgt für die nötigen Speicherformate.

then" ab, denn wenn die Funktion „FolderitemAvailable" des „Drag Item" „true" meldet, haben wir eine Datei gefunden. Dann rufen wir per „lade obj.folderitem" unser Unterprogramm „lade" auf und übergeben ihm diese Dateireferenz als Parameter.

Wir wissen nun, dass wir mit dem „MoviePlayer"-Steuerelement Quicktime-Filme jeder Art abspielen können. Dabei ist das Format des Films weit gehend unerheblich, solange Quicktime es unterstützt. Darunter fallen sowohl die vielen „echten" Quicktime-Filme (Dateityp „MooV" oder Dateiendung „.mov"), als auch alle unterstützten Fremdformate, wie zum Beispiel „AVI"- oder „MPEG"-Filme. Ein Film muss aber nicht zwingend einen Videokanal haben, sondern kann auch aus einer reinen Audiokomponente bestehen. Das hilft uns, um etwa die beliebten MP3-Dateien abzuspielen. Diese sind genau genommen nichts anderes als Filme ohne Bildkomponente.

WEITERE IDEEN: STANDBILDER MIT QUICKTIME LADEN

Mit Hilfe von Quicktime können wir auch Standbilder in vielen Formaten laden und darstellen, indem wir uns ein „FolderItem" (beispielsweise über die Funktion „open asfolderitem") zu der Bilddatei besorgen. Die Zeile „dim bild as picture" reserviert eine Variable für ein Bild, die anschließend mit „bild=datei.openaspicture" gefüllt wird. Auch hier bekommen wir ein „nil" zurück, wenn die Datei kein Bild enthält. Ansonsten können wir mit dem Bild genauso umgehen, wie mit denen im Projektfenster (als Beispiel: „backdrop=bild").

Eine Liste der über 30 Formate, die Quicktime unterstützt, finden Sie unter http://www.apple.com/quicktime/authoring/fileformats.html.

STREAMING MIT REALBASIC: FILME ÜBER DAS INTERNET

Seit der Version 4 gibt es in Quicktime die Möglichkeit, Filme als Streams live über das Internet anzusehen. Realbasic unterstützt dies ebenfalls und bietet die Funktion „Open URLMovie". Damit öffnet man eine Internet-URL als Quicktime-Film und kann den Film anschließend genau so wie in unserem Beispiel in einem Fenster anzeigen. Als Parameter muss man lediglich die exakte Internet-URL angeben „film=OpenURLMovie("http://www.apple.com/quicktime/showcase/live/bbc")". Um den Rest kümmert sich wie immer Quicktime selbst. Wenn Sie unseren Movieplayer noch erweitern und diese Funktionen integrieren, können Sie ihn schon als vollständigen Ersatz für den von Apple mitgelieferten Quicktime-Movieplayer verwenden.

BILDER KONVERTIEREN LEICHT GEMACHT: GRAFIKEN EXPORTIEREN

In Realbasic kann man auch einfach Bilder in anderen Formaten speichern und so einen einfachen Bildkonverter aufbauen. Dazu dient der Befehl „ExportPicture", der viele Bildformate zur Verfügung stellt. Dabei übergeben wir der Funktion lediglich das Bildobjekt. Der Benutzer darf sich dann in einem Speicherdialog selbst aussuchen, in welchem Format er das Bild gerne gesichert hätte. Auf dieser Seite oben rechts sehen Sie ein kleines Programmcodebeispiel dazu.

TAUSENDSASSA QUICKTIME: MUSIK MACHEN MIT MIDI-INSTRUMENTEN

Was wir bislang noch nicht angesprochen haben, ist die Möglichkeit, die in Quicktime integrierten Midi-Instrumente in Aktion zu hören. Quicktime bietet 128 verschiedene Instrumente, von denen sich 32 gleichzeitig mit einem Tonumfang von 128 Tönen spielen lassen. Realbasic unterstützt auch dies.

Zudem bietet Realbasic die Möglichkeit, Quicktime-Filme zu bearbeiten. Das fängt bei MP3-Dateien an, bei denen man den Interpreten und den Titel des Stücks anzeigen und ändern kann, und geht bis hin zu einem aus einzelnen Bilddateien selbst erzeugten und animierten Quicktime-Film.

Für Diaschauen oder eigene Filme liefert Realbasic in Verbindung mit Quicktime auch eine Vielzahl von Überblendeffekten, wie man sie von iMovie her kennt.

FAZIT

Wegen der guten Resonanz erweitern wir die Realbasic-Serie um einen fünften Teil. Im nächsten Monat werden wir uns ausführlich mit der Spieleprogrammierung beschäftigen und wollen lernen, wie man mit der integrierten Sprite-Engine umgeht. *cm*

Serie Realbasic

1 Einführung ..... **Heft 9/2000**
2 Taschenrechner im Eigenbau ..... **Heft 10/2000**
3 Grafikprüfung ..... **Heft 11/2000**
4 Quicktime-Programmierung ..... **Heft 12/2000**
5 Sprites und Spiele ..... **Heft 01/2001**