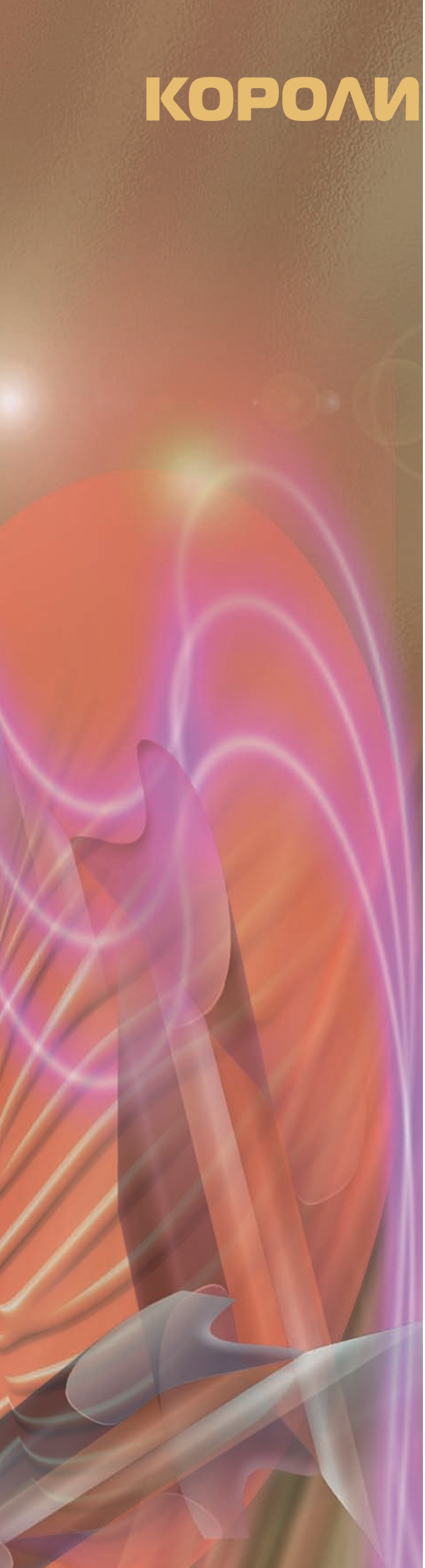
КОРОЛИ И КАПУСТА

В. Антуфьев
начальник сектора информационного обеспечения
Главного государственного экспертного центра
оценки образования,
О. Соловей
начальник IT-отдела ООО «Редтех»



Как вы относитесь к живописи французских импрессионистов конца XIX века? И, кстати, не кажется ли вам, что в картинах «розового» периода Пабло Пикассо слишком много розового?

Нет, с глазами у вас все в порядке, а с головами все в порядке у нас, в руках вы держите «Алгоритм безопасности» № 5, и до сих пор мы продолжаем разговор о принципах создания программного обеспечения систем видеонаблюдения.

В предыдущем номере мы обещали рассказать о внедрении в наше приложение простейшего программного детектора движения. А еще, как изменять размер окна видеозахвата программно, а не через диалог. Как отобразить в окне приложения видеоряд, не сохраняя последовательные кадры на жесткий диск для последующей вставки в контейнер графических изображений. Наконец, как записать интересующие нас события в AVI-файл.

Вообщем, по выражению великого О'Генри, мы обещали вам все, кроме королей и капусты.

В отличие от «народных избранников», мы свои обещания привыкли выполнять, поэтому придется продолжать нелегкий писательский труд.

Поговорим о прекрасном. То есть о лицезрении себя любимого на экране монитора и о том, как раскрасить этот шедевр во все цвета радуги в моменты, когда вам хочется почесать за ухом.

В принципе, все, что нам нужно для детектора движения, это взять два последовательных кадра видеоряда, попиксельно сравнить их и, в случае обнаружения различий, сгенерировать соответствующий сигнал. Например, закрасить зоны различия красным цветом. Или зеленым. На этом статью можно было бы и закончить, если бы не детали.

И начнем мы издалека.

Зачем компьютеру ток?

Вы знаете, как работает компьютер?

Почему-то бытует широко распространенное заблуждение, что компьютер – ужасно умное устройство. Как бы не так. Железяка не может быть умной по определению, если это не Железный дровосек из сказки, да и того по сюжету сделали из живого человека. Компьютер понимает только две вещи: «ноль» и «единицу». Скажем больше – он даже этого не понимает. Для каждого элемента компьютера существуют два состояния – «есть ток» и «нет тока». «Как бы умным» компьютер делают невообразимое количество этих самых элементов и человек, заставляющий элементы группироваться в определенные последовательности.

Для примера. Проведем небольшой эксперимент: отрежем сетевой шнур от торшера, вставим вилку шнура в розетку и на 10 секунд возьмемся руками за оголенные концы проводов.

Поздравляем, вы только что получили по двухканальному сетевому интерфейсу 125 байт информации о том, как работает компьютер.

Расчет здесь очень простой: компьютер понимает только положительные импульсы, значит, по двум проводам мы получим 100 импульсов в секунду – при частоте промышленного тока 50 Гц. Один импульс – 1 бит информации, за 10 секунд – 1000 бит. В байте – 8 бит, остальное посчитаете сами.

А если дать провода, скажем, всему личному составу мотострелковой дивизии, то это получится не то чтобы компьютер, но простенький калькулятор начала 90-х годов прошлого века. Только где же найти столько торшеров?

Точно так же, как для логической схемы компьютера основой является электрический импульс, в основе любого устройства графического вывода информации, будь то принтер, монитор, или что-нибудь еще, лежит точка. То есть все те красивости, которые вы наблюдаете на экране монитора, для компьютера – не более чем набор точек с различными комбинациями цвета и яркости. Компьютер не отличит «Мону Лизу» от ваших потуг в рисовании, сделанных в Point или Photoshop, но он может сообщить вам, что цветовая гамма точки с известными координатами изменилась по сравнению с предыдущим состоянием.

А нам, в принципе, только это и надо.

Сделав это эпохальное открытие, перейдем непосредственно к программированию, предварительно оговорив некоторые моменты.

1. Здесь не церковно-приходская школа. Повторять пройденное не будем. Поэтому, если вы не читали первую статью цикла и поблизости нет распространителя «Алгоритма безопасности», срочно отправляйтесь по ссылке: http://www.algoritm.org и качайте электронную версию № 4 за 2006 год.
2. Мы будем не дополнять, а изменять приложение, созданное в первой статье, поэтому, чтобы не упустить суть повествования, рекомендуем все-таки создать программу, описанную нами ранее, и затем сделать копию, над которой мы и будем работать.
3. Приносим извинения за ошибки, вкравшиеся в программный код. Недоглядели. В процедурах обработки нажатия кнопок «Формат…» и «Камера…» исправьте значения переменных `mCapWnd` и `mCapWnd1` на `mCapHwnd` и `mCapHwnd1` соответственно. Диалоговые окна формата видеозахвата и ресурсов камеры заработают.

Избавляемся от графических контейнеров

Вы спросите, зачем это нужно? Дело в том, что среда программирования VBA в Microsoft Office очень и очень неполноценная. Запихнуть картинку в графический контейнер мы можем, а вот каким-то образом обработать изображение – никак.

Поэтому заходим в окно графического проектирования форм (щелкаем по надписи `UserForm1`), выделяем контейнер Image1, закрываем глаза и жмем кнопку Del на клавиатуре. Производим ту же процедуру с контейнером Image2.

Страшно? Но необходимо.

Если вы помните, мы уже создали два окна видеозахвата, но в первой статье мы писали, что, поскольку Excel дает возможность получить идентификатор только главного окна приложения, в нем мы и разместили два невидимых окна.

Значит, все, что нам требуется – получить идентификатор самой формы `UserForm1`, разместить на месте контейнеров окна видеозахвата и сделать их видимыми.

Вообще, существует очень хорошее правило: если какое-либо действие недоступно в данной системе программирования, но возможно в принципе – используйте Windows API.

Все вышеперечисленные действия мы произведем в процедуре инициализации формы, но вначале установим несколько новых переменных и вызовов системных API-функций.

В секции General Declarations с новой строки пишем:

```
Private Const WS_CHILD = &H40000000
Private Const WS_VISIBLE = &H10000000
Private hWnd As Long
Private Declare Function FindWindow Lib _
«user32.dll» Alias «FindWindowA»_
(ByVal lpClassName As String, _
ByVal lpWindowName As String) As Long
```

Объясним смысл написанного. Объявляем константы `WS_CHILD` и `WS_VISIBLE`, содержащие значения стиля создаваемого окна в числовом выражении (наследственное по отношению к родительскому окну и видимое соответственно).

Объявляем переменную `hWnd`, в которой будет храниться идентификатор окна, доступный для любой процедуры формы.

Объявляем API-функцию `FindWindow` с параметрами `lpClassName` (имя класса) и `lpWindowName` (имя окна), которая найдет нам нужное окно (в нашем случае `UserForm1`) по тексту его заголовка.

Переписываем процедуру `UserForm_Initialize` следующим образом:

```
Private Sub UserForm_Initialize()
hWnd = FindWindow(vbNullString, UserForm1.Caption)
TCap = False
mCapHwnd = capCreateCaptureWindow _
(«VideoCapture», WS_VISIBLE Or WS_CHILD, 10, 8, _
320, 240, hWnd, 0)
TCap1 = False
mCapHwnd1 = capCreateCaptureWindow _
(«VideoCapture2», WS_VISIBLE Or WS_CHILD, 370, 8, _
320, 240, hWnd, 0)
End Sub
```

Если сравнить написанное с предыдущим вариантом, вы поймете, что мы сделали. Мы разъясним только новые компоненты.

`hWnd = FindWindow(vbNullString, UserForm1.Caption)` – ищем окно, содержащее в заголовке название нашей формы и записываем его идентификатор в переменную hWnd. При этом в качестве имени класса передаем системную константу, означающую нулевую строку.

`mCapHwnd = capCreateCaptureWindow («VideoCapture», WS_VISIBLE Or WS_CHILD, _ 10, 8, 320, 240, hWnd, 0)` – создаем окно видеозахвата 320x240, но на этот раз стиль окна указываем наследственный и видимый, с координатами 10 и 8 пикселей, относительно края нашей формы, в качестве родительского окна указываем найденный ранее идентификатор формы (`hWnd`).

Ту же процедуру повторяем для второго окна видеозахвата, указав координаты по оси X – 370 пикселей.

Вы скажете, что в окне разработки графического интерфейса ничего не появилось? И будете абсолютно правы. Окна появятся только при запуске приложения.

Детектор движения

Все дальнейшие изменения мы будем проводить только для первого окна видеозахвата. Не имеет смысла по два раза повторять одни и те же процедуры. Для нас важен принцип, а процедуры для второго окна вы можете дописать и сами, по аналогии с первым.

Кроме того, поскольку большинство новых процедур будет взаимосвязано, будем писать программный код комплексно, по мере написания объясняя то или иное наше действие.

Переходим к детектору движения.

В Windows API существуют функции GetPixel, получающая цвет точки с заданными координатами, и SetPixel, устанавливающая цвет для точки. Проблема в том, что для работы с точками (пикселями), нам необходимо знать так называемый контекст устройства – handle device context. Для этого тоже имеется API-функция GetDC, но она требует указать идентификатор окна устройства, а как раз идентификаторы окон видеозахвата мы и получили в процедуре инициализации формы.

В секции `General Declaration`, каждый раз с новой строки:

```
Private Declare Function GetDC Lib «user32» _
(ByVal hWnd As Long) As Long
```

– ищем контекст устройства по идентификатору окна.

```
Private Declare Function GetPixel Lib «gdi32» _
(ByVal hdc As Long, ByVal x As Long, _
ByVal y As Long) As Long
```

– получаем цвет точки с координатами x и y в контексте устройства hdc.

```
Private Declare Function SetPixel Lib «gdi32» _
(ByVal hdc As Long, ByVal x As Long, _
ByVal y As Long, ByVal crColor As Long) As Long
```

– устанавливаем цвет crColor для точки с координатами x и y в контексте устройства hdc.

```
Private hdc As Long
```

– объявляем переменную, содержащую контекст устройства доступной для всех процедур внутри формы.

```
Private P(320, 240) As Long
```

– объявляем численный массив размерностью по числу пикселей окна видеозахвата для хранения цвета каждого пикселя.

```
Private POn(320, 240) As Boolean
```

– объявляем логический массив, состоящий из значений True и False, для записи координат пикселей, изменивших и не изменивших цвет.

```
Private Const Tolerance As Integer = 40
```

– константа, определяющая нижнюю границу разности цвета пикселей предыдущего и текущего кадров видеоряда. Изменяя это значение, вы можете регулировать нижний предел срабатывания детектора движения.

Наконец, сама функция детектора движения – ее мы будем вызывать при обработке нажатия кнопки «Старт». После любого оператора End Sub, с новой строки:

```
Function Detect()
For I = 0 To 319
For J = 0 To 239
c = GetPixel(hdc, I, J)
R = c Mod 256
G = (c \ 256) Mod 256
B = (c \ 256 \ 256) Mod 256
c2 = P(I, J)
R2 = c2 Mod 256
G2 = (c2 \ 256) Mod 256
B2 = (c2 \ 256 \ 256) Mod 256
If Abs(R - R2) < Tolerance And _
Abs(G - G2) < Tolerance And _
Abs(b - B2) < Tolerance Then
POn(I, J) = True
Else
P(I, J) = GetPixel(hdc, I, J)
SetPixel hdc, I, J, vbRed
POn(I, J) = False
End If
Next J
Next I
For I = 1 To 318
For J = 1 To 238
If POn(I, J) = False And _
POn(I, J + 1) = False And _
POn(I, J - 1) = False And _
POn(I + 1, J) = False And _
POn(I - 1, J) = False Then
SetPixel hdc, I, J, vbGreen
End If
Next J
Next I
End Function
```

Разберем функцию построчно.

`For I = 0 To 319` – цикл, перебирающий все пиксели окна видеозахвата по горизонтали. Нижняя граница цикла – первый оператор Next I.

`For J = 0 To 239` – цикл, перебирающий все пиксели окна видеозахвата по вертикали. Нижняя граница цикла – первый оператор Next J.

`c = GetPixel(hdc, I, J)` – при каждой итерации считываем значение цвета очередного пикселя текущего кадра.

```
R = c Mod 256
G = (c \ 256) Mod 256
B = (c \ 256 \ 256) Mod 256
```

– выделяем численные значения красного (`R`), зеленого (`G`), синего (`B`) цветовых каналов для очередного пикселя как цельночисленный остаток деления на 256. Достаточно сложно для восприятия неспециалистом. Просто поверьте, что так оно и есть.

```
c2 = P(I, J)
R2 = c2 Mod 256
G2 = (c2 \ 256) Mod 256
B2 = (c2 \ 256 \ 256) Mod 256
```

– повторяем вышеописанную процедуру для предыдущего кадра. Значение цвета пикселя берем из массива `P()`.

```
If Abs(R - R2) < Tolerance And Abs(G - G2) < Tolerance And Abs(b - B2) < Tolerance Then
```

– условие, нижняя граница – оператор `End If`. Если абсолютное значение разности каждого цветового канала текущего и предыдущего кадров меньше нижнего предела срабатывания, тогда

`POn(I, J) = True` – для пикселя с координатами `I` и `J` заносим значение «истина» в массив `POn()`.

`Else` – иначе

`P(I, J) = GetPixel(hdc, I, J)` – заносим в массив `P()` значение цвета очередного пикселя текущего кадра.

```
SetPixel hdc, I, J, vbRed
```

– меняем цвет пикселя на красный.

`POn(I, J) = False` – для пикселя с координатами `I` и `J` заносим значение «ложь» в массив `POn()`.

В этой части функции мы определили все пиксели, изменившие цвет на величину больше предела срабатывания.

Но для исключения ложных срабатываний нам необходимо выделить из общей массы истинное движение. Поэтому вновь запускаем цикл:

```
For I = 1 To 318
For J = 1 To 238
```

– на этот раз исключаем из цикла пиксели, находящиеся по краям картинки.

```
If POn(I, J) = False And POn(I, J + 1) = False
And _
POn(I, J - 1) = False And POn(I + 1, J) = False
And _
POn(I - 1, J) = False Then
```

– если все прилежащие пиксели, включая исследуемый, изменили цвет, то

`SetPixel hdc, I, J, vbGreen` – меняем цвет исследуемого пикселя на зеленый.

Кстати, в этом месте можно посчитать общее количество пикселей, соответствующих истинному движению, для определения нижнего порога записи кадра в AVI-файл и выделить секторы, определять движение в которых нет необходимости (например, настроить детектор только на дверной проем и не учитывать прочее движение). Мы эти функции описывать не будем.

Предустановка формата окна видеозахвата

Вы, наверное, заметили, что при каждом новом запуске приложения формируются видеоокна того размера, который вы установили при предыдущем запуске?

Нам это тоже не нравится, поэтому попробуем установить формат окна программно.

Для предустановки формата окна видеозахвата используется пара системных сообщений `WM_CAP_GET_VIDEOFORMAT` и `WM_CAP_SET_VIDEOFORMAT`. Первым сообщением мы получаем структуру, содержащую параметры видеоформата, и заполняем ее данными по умолчанию. Далее мы можем изменить некоторые или все параметры и применить их к окну видеозахвата вторым системным сообщением. Здесь мы впервые сталкиваемся с понятием «структуры данных». Это не число и не строка, а некий упорядоченный набор параметров, и для того чтобы его получить, нам необходимо ввести специализированные API-функции и новый пользовательский тип данных.

Начинаем.

В секции `General Declarations`:

```
Private Declare Function SendMessageAsAny Lib _
«user32» Alias «SendMessageA» _
(ByVal hWnd As Long, ByVal wMsg As Long, _
ByVal wParam As Long, ByRef lParam As Any) As
Long
```

Все то же самое, только последний параметр функции мы передаем не по ссылке, а по значению и формат данных – `Any`, т.е. попросту «другой».

```
Private Const WM_CAP_GET_VIDEOFORMAT As Long =
WM_CAP_START + 44
Private Const WM_CAP_SET_VIDEOFORMAT As Long =
WM_CAP_START + 45
```

– присваиваем численные значения системным сообщениям.

Далее переходим в программный модуль Module1:

В секции `General Declarations` модуля:

```
Public Type BITMAPINFOHEADER
   biSize As Long
   biWidth As Long
   biHeight As Long
   biPlanes As Integer
   biBitCount As Integer
   biCompression As Long
```

```
    biSizeImage As Long
    biXPelsPerMeter As Long
    biYPelsPerMeter As Long
    biClrUsed As Long
    biClrImportant As Long
End Type
Public Type BITMAPINFO
    bmiHeader As BITMAPINFOHEADER
    bmiColors() As Long
End Type
```

Объявляем пользовательский тип данных `BITMAPINFOHEADER`, объявляем переменные параметров формата окна внутри типа. Затем объявляем еще один пользовательский тип `BITMAPINFO`, одной из переменных которого является ранее объявленный тип `BITMAPINFOHEADER`. В принципе, нам нужны только четыре параметра, но структуру мы обязаны объявить полностью.

Далее переходим в окно программного кода `UserForm1`.

В процедуре обработки нажатия кнопки «Старт» первого окна видеозахвата `CommandButton1_Click()`, сразу после начала процедуры, с новой строки:

```
Dim BmpFormat As BITMAPINFO
```

– объявляем переменную `BmpFormat` для хранения структуры данных формата видеоокна.

```
SendMessageAsAny mCapHwnd, _
WM_CAP_GET_VIDEOFORMAT, Len(BmpFormat), BmpFormat
```

– посылаем системное сообщение получить структуру формата видеоокна. Последовательно указываем идентификатор окна `(mCapHwnd)`, само сообщение `(WM_CAP_GET_VIDEOFORMAT)`, длину структуры `(Len(BmpFormat))` и, наконец, саму структуру `(BmpFormat)`. Структура заполняется данными, установленными по умолчанию.

```
BmpFormat.bmiHeader.biHeight = 320
```

– изменяем размер окна по горизонтали на 320 пикселей.

```
BmpFormat.bmiHeader.biWidth = 240
```

– изменяем размер окна по вертикали на 240 пикселей.

```
BmpFormat.bmiHeader.biBitCount = 12
```

– устанавливаем глубину цвета 12 бит на пиксель.

```
BmpFormat.bmiHeader.biSizeImage = _
BmpFormat.bmiHeader.biHeight * _
BmpFormat.bmiHeader.biWidth * _
(BmpFormat.bmiHeader.biBitCount / 8)
```

– устанавливаем размер картинки в байтах (длина * ширина * (бит на пиксель / 8)).

```
SendMessageAsAny mCapHwnd, _
WM_CAP_SET_VIDEOFORMAT, Len(BmpFormat), BmpFormat
```

– посылаем системное сообщение применить настройки формата. Передаваемые параметры – такие же, как и в первом сообщении.

Теперь, как бы мы не меняли размеры видеоокна, при нажатии кнопки «Старт» появится окно размером 320x240 пикселей с форматом изображения I420 (если надо установить формат изображения RGB 24, то параметр `biBitCount` должен быть равным 24). Не очень сложно, правда?

Мотор!!!

И, наконец, одна из самых важных для видеонаблюдения функций – запись видеоряда в файл.

Мы будем использовать не потоковое видео, а запись кадра по событию. Событием в данном случае послужит сам захват кадра. Для своих целей вы можете использовать любые другие события. Срабатывание детектора движения, например.

Введем в секцию `General Declarations` формы `UserForm1` объявления новых системных сообщений.

```
Private Const WM_CAP_DLG_VIDEOCOMPRESSION _
As Long = WM_CAP_START + 46
```

– открывает диалог настройки сжатия видеопотока.

```
Private Const WM_CAP_FILE_SET_CAPTURE_FILE _
As Long = WM_CAP_START + 20
```

– устанавливает файл для захвата видеопотока (по умолчанию – C:\CAPTURE.AVI).

```
Private Const WM_CAP_SINGLE_FRAME_OPEN _
As Long = WM_CAP_START + 70
```

– открыть ранее установленный файл для захвата одиночных кадров.

```
Private Const WM_CAP_SINGLE_FRAME_CLOSE _
As Long = WM_CAP_START + 71
```

– закрыть ранее открытый видеофайл (повторное открытие файла для захвата уничтожает все данные).

```
Private Const WM_CAP_SINGLE_FRAME _
As Long = WM_CAP_START + 72
```

– записать одиночный кадр в файл.

Кроме того, нам потребуется ввести еще один пользовательский тип данных для настройки параметров видеозахвата.

В секции General Declarations модуля Module1 запишем следующее:

```
Public Type CAPTUREPARMS
    dwRequestMicroSecPerFrame As Long
    fMakeUserHitOKToCapture As Long
    wPercentDropForError As Long
    fYield As Long
    dwIndexSize As Long
    wChunkGranularity As Long
    fUsingDOSMemory As Long
    wNumVideoRequested As Long
    fCaptureAudio As Long
    wNumAudioRequested As Long
    vKeyAbort As Long
    fAbortLeftMouse As Long
    fAbortRightMouse As Long
    fLimitEnabled As Long
    wTimeLimit As Long
    fMCIControl As Long
    fStepMCIDevice As Long
    dwMCIStartTime As Long
    dwMCIStopTime As Long
    fStepCaptureAt2x As Long
    wStepCaptureAverageFrames As Long
    dwAudioBufferSize As Long
    fDisableWriteCache As Long
    AVStreamMaster As Long
End Type
```

И опять из этой большой структуры нам понадобится всего четыре параметра, поэтому не будем останавливаться на этом подробно.

В принципе, если позволит объем, то в третьей статье цикла мы приведем указания всех системных сообщений с передаваемыми параметрами и описание всех необходимых структур.

А пока переходим к форме `UserForm1`.

В окне графического интерфейса переименовываем кнопку «Формат1» на «Запись», а «Камера1» на «Сжатие».

Процедуру обработки нажатия кнопки «Запись» переписываем следующим образом:

```
Private Sub CommandButton3_Click()
Dim capParms As CAPTUREPARMS
```

– объявляем переменную capParms для хранения параметров видеозахвата.

```
SendMessageAsAny mCapHwnd,
WM_CAP_GET_SEQUENCE_SETUP, Len(capParms), capParms
```

– системное сообщение получить параметры захвата по умолчанию.

```
capParms.fAbortLeftMouse = False
```

– отключить остановку видеозахвата кликом левой кнопки мыши.

```
capParms.fAbortRightMouse = False
```

– отключить остановку видеозахвата кликом правой кнопки мыши.

```
capParms.fMakeUserHitOKToCapture = False
```

– отключить начало захвата по нажатию кнопки ОК во всплывающем окне.

```
capParms.fYield = True
```

– параметр, ради которого все это и затевалось. Захватывать видео в файл в фоновом режиме. Если параметр равен `False` (по умолчанию), на период записи все приложение блокируется.

```
SendMessageAsAny mCapHwnd,
WM_CAP_SET_SEQUENCE_SETUP, Len(capParms), cap-
Parms
```

– применить настройки видеозахвата.

```
If CommandButton3.Caption = «Запись» Then
```

– если надпись на кнопке – «Запись»

```
CommandButton3.Caption = «Стоп»
```

– сменить надпись кнопки на «Стоп»

```
sFileName = ThisWorkbook.Path & «\» & _
CStr(Format(Date, «dd.mm.yy»)) & «_» & _
CStr(Format(Time, «hh.mm.ss»)) & «.avi»
```

– присвоить видеофайлу имя: каталог приложения\дата_время.avi (так, например, новогодний файл будет называться «31.12.07_23.59.59.avi»).

```
SendMessageAsString mCapHwnd, _
WM_CAP_FILE_SET_CAPTURE_FILE, 0, sFileName
```

– системное сообщение установить файл для видеозахвата.

```
SendMessageAsLong mCapHwnd, _
WM_CAP_SINGLE_FRAME_OPEN, 0, 0
```

– системное сообщение открыть файл для захвата одиночных кадров.

```
ElseIf CommandButton3.Caption = «Стоп» Then
```

– иначе, если надпись на кнопке – «Стоп», тогда

```
CommandButton3.Caption = «Запись»
```

– сменить надпись кнопки на «Запись»,

```
SendMessageAsLong mCapHwnd, _
WM_CAP_SINGLE_FRAME_CLOSE, 0, 0
```

– системное сообщение закрыть видеофайл.

```
End If
End Sub
```

Переписываем процедуру нажатия кнопки «Сжатие»:

```
Private Sub CommandButton4_Click()
SendMessageAsLong mCapHwnd, _
WM_CAP_DLG_VIDEOCOMPRESSION, 0, 0
End Sub
```

– системное сообщение открыть диалоговое окно настроек видеокомпрессии.

Наконец, переписываем процедуры обработки нажатия кнопок «Старт» первого и второго окон видеозахвата. Комментировать будем только новые элементы:

```
Private Sub CommandButton1_Click()
Dim BmpFormat As BITMAPINFO
SendMessageAsAny mCapHwnd, _
WM_CAP_GET_VIDEOFORMAT, Len(BmpFormat), _
BmpFormat
BmpFormat.bmiHeader.biHeight = 320
BmpFormat.bmiHeader.biWidth = 240
BmpFormat.bmiHeader.biBitCount = 12
BmpFormat.bmiHeader.biSizeImage = _
BmpFormat.bmiHeader.biHeight * _
BmpFormat.bmiHeader.biWidth  * _
(BmpFormat.bmiHeader.biBitCount / 8)
SendMessageAsAny mCapHwnd, _
WM_CAP_SET_VIDEOFORMAT, Len(BmpFormat), BmpFor-
mat
TCap = True
SendMessageAsLong mCapHwnd, _
WM_CAP_DRIVER_CONNECT, 0, 0
Do While TCap = True
For I = 1 To 1000
DoEvents
If I = 1000 Then
SendMessageAsLong mCapHwnd, WM_CAP_GRAB_FRAME,
0, 0
If TCap = True Then
```

– все выше написанное комментировалось ранее.

```
If CommandButton3.Caption = «Стоп» Then
```

```
SendMessageAsLong mCapHwnd, WM_CAP_SINGLE_FRAME,
0, 0
End If
```

– если название кнопки CommandButton3 – «Стоп», тогда захватить одиночный кадр в файл.

```
hdc = GetDC(mCapHwnd)
```

– получить контекст устройства первого окна видеозахвата.

```
Call Detect
```

– вызвать функцию `Detect()` детектора движения для текущего кадра.

```
End If
If TCap1 = True Then
SendMessageAsLong mCapHwnd1, WM_CAP_GRAB_FRAME,
0, 0
End If
End If
Next
Loop
End Sub
```

Процедура обработки нажатия кнопки «Старт» второго окна, надеемся, в комментариях не нуждается:

```
Private Sub CommandButton7_Click()
TCap1 = True
SendMessageAsLong mCapHwnd1, _
WM_CAP_DRIVER_CONNECT, 0, 0
Do While TCap1 = True
For I = 1 To 1000
DoEvents
If I = 1000 Then
SendMessageAsLong mCapHwnd1, WM_CAP_GRAB_FRAME,
0, 0
If TCap = True Then
SendMessageAsLong mCapHwnd, WM_CAP_GRAB_FRAME,
0, 0
If TCap = True Then
If CommandButton3.Caption = «Стоп» Then
SendMessageAsLong mCapHwnd, WM_CAP_SINGLE_FRAME,
0, 0
End If
hdc = GetDC(mCapHwnd)
Call Detect
End If
End If
End If
Next
Loop
End Sub
```

Если вы заметили, мы убрали из этих процедур сохранение изображения в файл BMP. Поскольку окна видеозахвата теперь видимые, нам больше это не нужно.

Вот, в общем, и все.

Жмите в меню «Debug» – «Compile VBA Project», сохраняйте, запускайте приложение и наслаждайтесь. Только не забудьте подключить видеокамеры. Хотя бы одну.

Заключение

Надеемся, то, что мы написали, когда-нибудь пригодится вам в работе. Или хотя бы для расширения кругозора.

Не боясь показаться назойливыми, напомним вам еще раз: если что-то не получается или лень перепечатывать букву за буквой, пишите: antufjev@yandex.ru. Поможем, чем сможем, или пришлем готовый файл.

Ну, и традиционное…

До встречи в следующем номере!