



**ПРОГРАММНЫЙ МОДУЛЬ ДЛЯ
ОТОБРАЖЕНИЯ И ОБРАБОТКИ
ИЗОБРАЖЕНИЙ**

Справочное руководство

Версия 1.0

120-039



НАУЧНО-ТЕХНИЧЕСКИЙ ЦЕНТР

Module® и **NeuroMatrix®** являются зарегистрированными торговыми марками ЗАО НТЦ "Модуль". Все другие торговые марки являются исключительной собственностью их соответствующих владельцев.

Содержание

1. Общие сведения	5
2. Использование VShell	6
2.1. Подключаемые модули и заголовочные файлы	6
2.2. Алгоритм работы VShell	6
2.3. Структура пользовательской программы	7
2.4. Описание примеров	8
2.4.1. AviPlayer	8
2.4.2. SmoothFilter	8
2.4.3. VectorGraph	8
3. Программная оболочка	9
3.1. Оконный интерфейс	9
3.1.1. Панель инструментов приложения	11
3.1.2. Меню приложения	13
3.1.3. Быстрые клавиши	15
4. Описание функций, структур и идентификаторов VShell	16
4.1. Функции VShell	16
4.1.1. Функции инициализации и организации цикла обработки изображений	16
4.1.2. Функции получения и установки данных изображения	23
4.1.3. Функции получения параметров изображения	25
4.1.4. Функции для работы с палитрами	27
4.1.5. Функции для работы с элементами управления	29
4.1.6. Функции векторной графики	36
4.2. Структуры VShell	59
4.2.1. S_VS_MouseStatus	59
4.2.2. S_VS_Pal	59
4.2.3. S_VS_Point	59
4.2.4. S_VS_PointF	60
4.3. Идентификаторы VShell	60
4.3.1. Идентификатор групповой отрисовки	60
4.3.2. Идентификатор окна - источника изображений	60
4.3.3. Идентификатор отсутствия цвета	60
4.3.4. Идентификаторы курсоров мыши	60
4.3.5. Идентификаторы отображения векторов движения	61
4.3.6. Идентификаторы цветов	61
4.3.7. Идентификаторы состояния мыши	62
4.3.8. Идентификаторы состояния программы	62
4.3.9. Идентификаторы стиля пера для векторной графики	62
4.3.10. Идентификаторы типов изображений	63
4.3.11. Максимальные значения	64
4.3.12. Тип для задания цвета для custom палитры	64
5. Ограничения в демонстрационной версии	65

Список иллюстраций

3.1.	10
3.2.	12
4.1.	28

1. Общие сведения

Программный модуль VShell работает в среде Windows (95/98/NT/2000/XP), является библиотекой динамической загрузки и служит для обработки и последующего вывода изображений. VShell адресован, прежде всего, для разработчиков, занимающихся обработкой, исследованием и генерацией изображений. Модуль предоставляет пользователю удобный многооконный интерфейс для вывода изображений, позволяет разработчику сконцентрироваться на выполнении целевой задачи и избавляет его от рутинной работы, связанной с выводом графики средствами GDI, создания окон и обработки сообщений.

VShell поддерживает операции, связанные с загрузкой и сохранением изображений (или последовательности изображений), производит масштабирование, навигацию, отображение сетки, векторов движения, а также позволяет легко создавать векторные изображения.

VShell предоставляет пользователю интерфейс для работы с такими элементами управления, как окна ввода, выключатели, группы переключателей и слайдеры.

Функции библиотеки можно вызывать как из консольных приложений (из функции main), так и оконных приложений (из функции WinMain).

2. Использование VShell

2.1. Подключаемые модули и заголовочные файлы

Для работы с программным модулем необходимо включить файл VShell.dll в область "видимости" операционной системы. Наилучшим решением будет создание переменной среды окружения, например, с именем VSHELL, в которой записывается путь к каталогам дистрибутива и добавление в переменную окружения PATH строки %VSHELL%\bin.

Каталог дистрибутива содержит 5 подкаталогов:

- bin - содержит динамически подключаемую библиотеку VShell32.dll (для 32-х бтовой платформы) или VShell64.dll (для 64-х бтовой платформы) и примеры
- doc - содержит файл справки VShell.chm
- include - содержит заголовочный файл с описанием функций библиотеки
- lib - содержит подключаемый модуль для раннего связывания VShell32.lib (для 32-х бтовой платформы) или VShell64.lib (для 64-х бтовой платформы)
- examples - примеры использования библиотеки

Для использования библиотеки C/C++ программах включите в исходный файл директиву `#include "VShell.h"`. Если используется среда разработки VC++6.0 / 7.0, то для раннего связывания подключите к проекту файл VShell32.lib (VShell64.lib). Используйте переменную окружения VSHELL32 (VSHELL64) для задания пути к файлам VShell.h и VShell32.lib (VShell64.lib).

2.2. Алгоритм работы VShell

Для эффективного использования программного модуля необходимо понимать принцип работы VShell. Обработка и вывод изображений происходит итеративно, по одному кадру за итерацию. Одна итерация включает в себя загрузку изображения (если необходимо обрабатывать ранее созданное изображение), собственно обработку изображения и вывод изображения в заданное окно. Перед запуском цикла обработки необходимо инициализировать библиотеку и создать окна для вывода изображений.

Для загрузки изображения необходимо задать источник, то есть путь к изображению (bmp файл), к последовательности изображений, или к AVI файлу. VShell поддерживает работу с растровыми изображениями с различной цветовой организацией: 1, 4, 8, 16, 24 и 32 бит на пиксель.

При работе с последовательностью изображений необходимо задать путь и имя первого файла в последовательности. Имена файлов в последовательности должны формироваться из префикса (необязательно) и числа. Все числа должны последовательно возрастать и должны состоять из одинакового количества цифр. Максимальное количество цифр в числе - 6. Например, последовательность может выглядеть следующим образом: Img0000.bmp, Img0001.bmp, Img0002.bmp и так далее. Последовательность изображений обрабатывается циклически, то есть после загрузки последнего изображения вновь будет загружено первое изображение. Если в имени

файла источника отсутствует число, или последовательность прерывается после первого изображения, то в каждой итерации будет загружаться одно и тоже изображение.

При работе с AVI файлом следует указать его имя. Здесь также используется циклическая обработка, как и при работе с последовательностью растровых изображений. Следует учитывать, что при чтении AVI файла используются вызовы Video For Windows (VFW), поэтому декодер AVI должен быть совместим с VFW.

Источник изображения можно указать в программе с помощью вызова [VS_Bind\(\)](#).

Если предполагается генерация новых изображений, то указание источника не требуется - достаточно создать окно с требуемыми характеристиками и выводить в него буфер с сформированным изображением.

В процессе обработки изображений можно на них накладывать векторную графику. Векторная графика всегда накладывается на изображение. Элементы векторной графики выводятся в том порядке, в котором их создавал разработчик.

2.3. Структура пользовательской программы

Рассмотрим структуру программы на следующем примере:

```
#include "VShell.h" void main() { unsigned char *pbOriginal; unsigned
char *pbWork; S_VS_Pal pPal[256]; if(!VS_Init()){ return 0; } if(!
VS_Bind("Test.avi")){ return 0; } VS_GetPalette(VS_SOURCE,
pPal); VS_CreateImage("Original", 1, VS_GetWidth(VS_SOURCE),
VS_GetHeight(VS_SOURCE), VS_GetType(VS_SOURCE),
pPal); VS_CreateImage("Work", 2, VS_GetWidth(VS_SOURCE),
VS_GetHeight(VS_SOURCE), RGB24, NULL); pbOriginal = new unsigned
char[VS_GetBufSize(1)]; pbWork = new unsigned char[VS_GetBufSize(2)];
while(VS_Run()){ VS_GetData(VS_SOURCE, pbOriginal); VS_SetData(1,
pbOriginal); VS_GetBGData(VS_SOURCE, pbWork); Work(pbWork,
VS_GetWidth(VS_SOURCE), VS_GetHeight(VS_SOURCE)); VS_SetData(2,
pbWork); VS_Draw(VS_DRAW_ALL); } delete [] pbOriginal; delete []
pbWork; }
```

Для работы с функциями библиотеки необходимо подключить заголовочный файл и вызвать функцию инициализации ([VS_Init\(\)](#)). Далее вызывается функция [VS_Bind\(\)](#) для задания источника изображений. Здесь источником изображений является AVI файл. Если связывание с источником прошло успешно, то можно создать окно для вывода источника ([VS_CreateImage\(\)](#)) с идентификатором 1. На этом этапе доступны данные об источнике изображений: ширина, высота, палитра, цветовая организация. При создании первого окна указываются такие же параметры, как у источника, в том числе передаётся и палитра источника. Далее, создаётся второе окно для вывода обработанного изображения. Пусть это изображение будет с определённой цветовой организацией, то есть цветное изображение, 24 бита на пиксель ([VS_RGB24](#)). Для каждого из изображений выделяется буфер. Размер необходимого буфера можно получить с помощью вызова [VS_GetBufSize\(\)](#).

Далее запускается цикл обработки изображения. При этом Функция [VS_Run\(\)](#) заблокирует работу приложения до тех пор, пока пользователь не начнёт проигрывать AVI файл. В цикле происходит следующее:

- Копирование визуальных данных кадра в буфер pbOriginal ([VS_GetData\(\)](#)).
- Вывод кадра в первое окно ([VS_SetData\(\)](#)).
- Копирование визуальных данных кадра в буфер pbWork с преобразованием в [VS_RGB24](#) ([VS_GetBGRData\(\)](#)).
- Вызывается некая функция обработки изображения Work() (здесь на приводится).
- Вывод результата обработки во второе окно ([VS_SetData\(\)](#)).

Последней операцией в цикле является отрисовка всех изображений ([VS_Draw\(\)](#)).

В цикле можно удалять и создавать окна, получить состояние мыши и элементов управления, наложить на изображение векторную графику и многое другое.


Рекомендуется при работе с VShell придерживаться такой структуры программы, хотя возможны и другие варианты. Например, если разработчику нужно сгенерировать новое изображение, то можно вообще не задавать источник изображений.

2.4. Описание примеров

2.4.1. AviPlayer

Пример демонстрирует создание простого проигрывателя последовательностей изображений и AVI файлов. В функцию [VS_Bind\(\)](#) передаётся NULL, что позволяет пользователю выбрать источник после запуска приложения. В примере создаётся окно для вывода изображений совместимое с источником.

2.4.2. SmoothFilter

Пример демонстрирует программу, в которой заданное изображение обрабатывается сглаживающим фильтром. В функции [VS_Bind\(\)](#) явно задаётся путь к изображению - источнику. В примере создаётся два окна: для вывода источника и результата обработки. Размер сглаживающего фильтра задаётся в окне ввода, поэтому в примере демонстрируется ещё и работа с элементом управления. Чтобы увидеть окно ввода нажмите кнопку  на панели инструментов. После ввода размера фильтра нажмите клавишу <ENTER> для того, чтобы показать VShell, что ввод окончен.

2.4.3. VectorGraph

Пример демонстрирует построение изображений с использованием векторной графики. Источник изображений не задаётся. Векторная графика накладывается на пустой буфер (на чёрном цвете). В примере для вывода графики используются 2 окна: в первое выводиться целочисленная векторная графика, во второе - графика с числами с плавающей точкой. Для того, чтобы увидеть особенности векторной графики с плавающей точкой необходимо увеличить изображение.

3. Программная оболочка

3.1. Оконный интерфейс

VShell предоставляет пользователю программную оболочку для вывода изображений. Программная оболочка является MDI приложением, в котором выводятся пользовательские изображения, каждое в своём дочернем окне. На рисунке представлен общий вид программной оболочки:

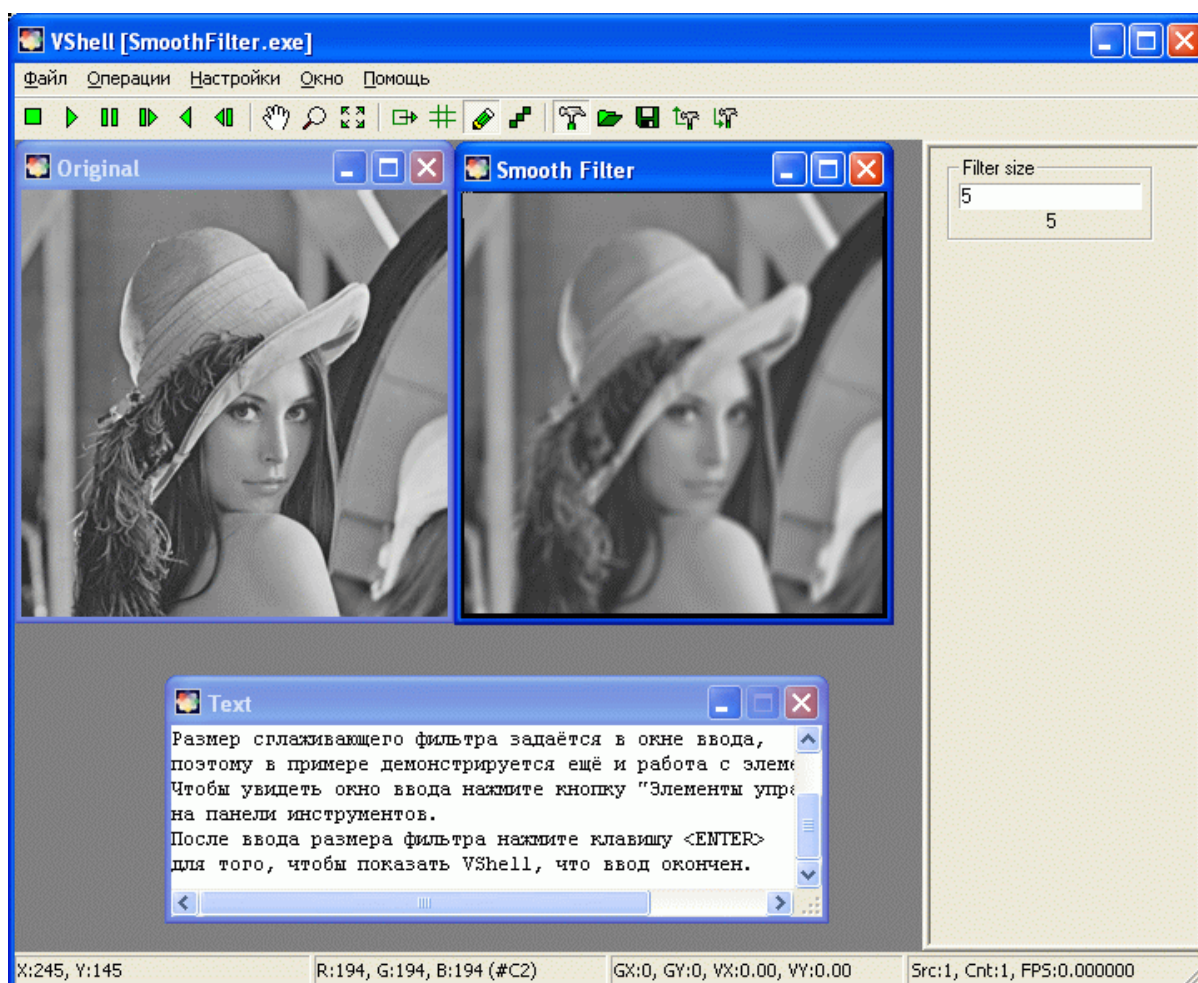



Рисунок 3.1 -


Здесь загружено два изображения в дочерние MDI окна: оригинальное изображение и результат работы сглаживающего фильтра. В верхней части окна находятся меню программы и панель инструментов. Для быстрого вызова некоторых элементов меню можно воспользоваться всплывающим меню, которое вызывается нажатием правой


кнопки мыши на изображении. Справа располагается специальное окно для размещения элементов управления. На рисунке показан один элемент управления - поле ввода для задания размера сглаживающего фильтра.


В нижней части окна приложения располагается строка состояния, которая разбита на 4 части. В первой части строки отображаются координаты курсора мыши в дочернем окне. Во второй части строки отображаются цветовые компоненты (RGB) пикселя, на который наведён курсор мыши, а также значение пикселя, взятое непосредственно из буфера изображения (представлено в шестнадцатеричной системе). В третью часть строки выводятся координаты ячейки сетки и координаты блока векторов движения, на который указывает курсор мыши. В четвёртую часть строки выводится номер текущего кадра источника изображений, общий счётчик кадров и количество кадров в секунду (FPS).


3.1.1. Панель инструментов приложения


Кнопка  сбрасывает счётчик кадров последовательности изображений. Если после проигрывания части последовательности нажать на эту кнопку, то изображения начнут загружаться снова, начиная с первого.


Кнопка  запускает цикл загрузки и отображения изображений в правильном порядке, начиная от первого. Последовательности и AVI файлы проигрываются циклически, то есть после отображения последнего кадра будет загружен первый и т. д.


Кнопка  отключает функцию загрузки следующего/предыдущего изображения. Если нажаты одновременно кнопки "Пауза" и "Запуск", то запустится только цикл отображения изображений, то есть всегда будет загружаться один и тот же кадр последовательности, без перехода к следующему/предыдущему кадру. Эта кнопка служит для того, чтобы можно было приостановить обработку последовательности изображений и сфокусироваться на обработке одного кадра.


Кнопка  запускает одну итерацию цикла загрузки и отображения следующего кадра (шаг вперёд).

Кнопка  запускает цикл загрузки и отображения изображений в обратном порядке, начиная от последнего кадра. Последовательности и AVI файлы проигрываются циклически, то есть после отображения первого кадра будет загружен последний и т. д.

Кнопка  запускает одну итерацию цикла загрузки и отображения предыдущего кадра (шаг назад).


Кнопка  позволяет перемещать изображение внутри окна. Используется для навигации по увеличенному изображению.


Кнопка  позволяет увеличить изображение. Для уменьшения увеличенного изображения удерживайте клавишу <SHIFT>

Кнопка  привязывает размер окна к размеру изображения. Используется при масштабировании изображений.

Кнопка  включает или выключает отображение векторов движения.

Кнопка  включает или выключает отображение сетки.

Кнопка  включает или выключает отображение пользовательской векторной графики. Особенность векторной графики состоит в том, что при масштабировании изображений её элементы сохраняют свои параметры, такие, например, как ширина линии, при этом все координаты векторной графики масштабируются вместе с изображением. Векторную графику строит сам пользователь с помощью функций библиотеки VShell.

Кнопка  включает или выключает растеризацию векторной графики. Этот эффект заметен при увеличении изображения на которое наложена векторная графика. На рисунке показана линия, наложенная на увеличенное в 16 раз изображение. Слева - растеризация выключена, справа - включена.

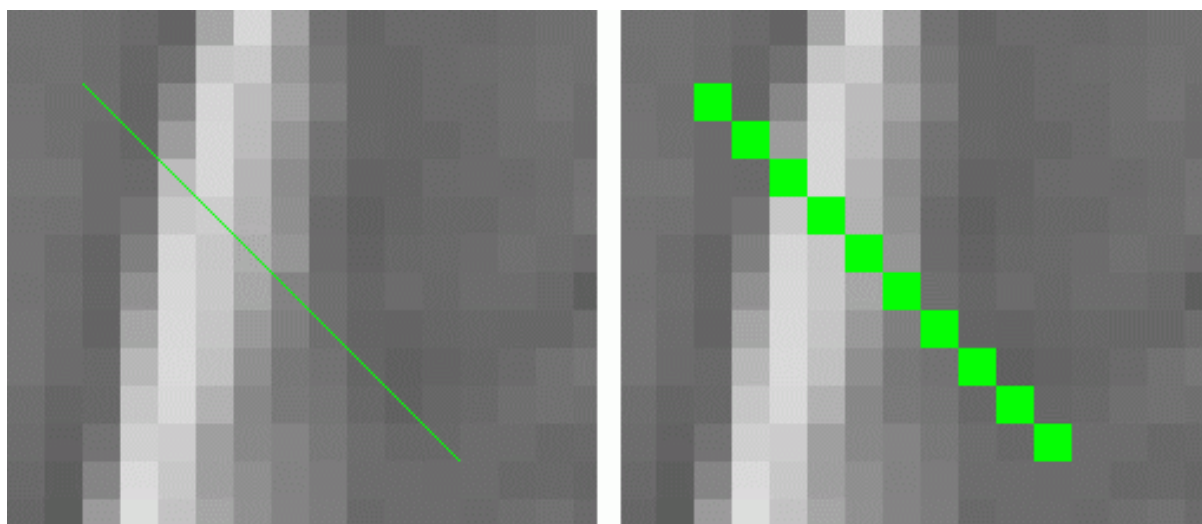




Рисунок 3.2 -

Кнопка  показывает или прячет специальное окно для элементов управления (выводится в правой части окна приложения).


Кнопка  сохраняет контекст VShell. При этом в конфигурационный файл записываются следующие параметры приложения:


Текущий размер и позиция главного окна приложения.

Текущий размер и позиция всех дочерних окон.

Состояние меню и панели инструментов.

Контекст VShell сохраняется в файле VShell.dat в той же директории, откуда стартовало приложение. При последующих запусках приложения контекст VShell восстанавливается в соответствии с сохранёнными параметрами. ВНИМАНИЕ: файл VShell.dat является бинарным файлом и его не рекомендуется модифицировать. Если требуется вернуться к параметрам по умолчанию, можно просто удалить этот файл.

Кнопка  сохраняет параметры элементов управления, такие как состояние слайдеров, переключателей, содержимое окон ввода и т. д. При этом требуется указать имя файла для сохранения этих параметров.

Кнопка  загружает параметры элементов управления, такие как состояние слайдеров, переключателей, содержимое окон ввода и т. д. При этом требуется указать имя файла для загрузки этих параметров.

3.1.2. Меню приложения

3.1.2.1. Файл

Сохранить изображение - сохранение отдельного изображения. При запущенном цикле опция не доступна, так как во время проигрывания последовательности не известно, какое именно изображение следует сохранить. Для сохранения изображения необходимо остановить проигрывание.

Сохранить последовательность - сохранение последовательности изображений в AVI файл, или в последовательность растровых изображений.

Сохранить контекст VShell - см. описание кнопки  на панели инструментов

Сохранить контекст эл. управления - см. описание кнопки  на панели инструментов

Загрузить контекст эл. управления - см. описание кнопки  на панели инструментов

Выход - выход из приложения.

3.1.2.2. Операции

Сброс - см. описание кнопки  на панели инструментов

Запуск вперёд - см. описание кнопки  на панели инструментов


Пауза - см. описание кнопки  на панели инструментов


Шаг вперёд - см. описание кнопки  на панели инструментов

Запуск назад - см. описание кнопки  на панели инструментов


Шаг назад - см. описание кнопки  на панели инструментов

Навигация - см. описание кнопки  на панели инструментов


Масштабирование - см. описание кнопки  на панели инструментов

Векторы - см. описание кнопки  на панели инструментов

Сетка - см. описание кнопки  на панели инструментов

Размер окна по изображению - см. описание кнопки  на панели инструментов

Растеризовать - см. описание кнопки  на панели инструментов

Векторная графика - см. описание кнопки  на панели инструментов

Переполнения - включение или выключение отображения переполнения. Переполнения отображаются только для изображений типа [RGB8_16](#) или [RGB8_32](#), при этом красным цветом подсвечиваются пиксели, значения которых выше 128 и синим цветом - пиксели со значениями меньшими, чем (-128).

3.1.2.3. Настройки


Векторы - задание цвета по умолчанию для отображения векторов движения.

Сетка - задание размера и цвета для отображения сетки.

Яркость/Контрастность - опция позволяет включить преобразование изображения непосредственно перед его выводом, и настроить такие параметры преобразования как яркость и контрастность. Значения яркости и контрастности находятся в диапазоне от -128 до +127 включительно. Для изображений типа [RGB8_8/RGB8_16/RGB8_32](#) вместо контрастности используется множитель. Параметры множителя можно задать в настройках (кнопка "настройка" под слайдером).

Сохранить последовательность кадров как... - опция позволяет определить имя файла в который будет сохраняться последовательность кадров. Если это имя не задано, то при каждой попытке сохранить последовательность будет выводиться окно выбора файла для сохранения.

3.1.2.4. Окно

Элементы управления - см. описание кнопки  на панели инструментов

3.1.2.5. Помощь

О программе - вызов окна с информацией о программе.


3.1.3. Быстрые клавиши


Alt+F1 - сохранение отдельного изображения. При запущенном цикле опция не доступна, так как во время проигрывания последовательности не известно, какое именно изображение следует сохранить. Для сохранения изображения необходимо остановить проигрывание.

Alt+F2 - сохранение последовательности изображений в AVI файл, или в последовательность растровых изображений.


Alt+F3 - сохранить контекст VShell (см. описание кнопки  на панели инструментов)

Alt+F4 - выход из приложения.


Ctrl+Home - сброс (см. описание кнопки  на панели инструментов)

Ctrl+Up - запуск вперёд (см. описание кнопки  на панели инструментов)

Pause - пауза (см. описание кнопки  на панели инструментов)

Ctrl+Right - шаг вперёд (см. описание кнопки  на панели инструментов)


Ctrl+Down - запуск назад (см. описание кнопки  на панели инструментов)

Ctrl+Left - шаг назад (см. описание кнопки  на панели инструментов)

F2 - навигация (см. описание кнопки  на панели инструментов)

F3 - масштабирование (см. описание кнопки  на панели инструментов)

F4 - размер окна по изображению (см. описание кнопки  на панели инструментов)

F5 - векторная графика (см. описание кнопки  на панели инструментов)

F6 - растеризовать (см. описание кнопки  на панели инструментов)

F7 - элементы управления (см. описание кнопки  на панели инструментов)

F1 - вызов окна с информацией о программе.

Alt++ - увеличение изображения. Размеры окна изменяются в соответствии с изображением

Alt+- - уменьшение изображения. Размеры окна изменяются в соответствии с изображением

4. Описание функций, структур и идентификаторов VShell

4.1. Функции VShell

4.1.1. Функции инициализации и организации цикла обработки

4.1.1.1. VS_Bind

Функция связывания источника с заданной последовательностью кадров или AVI файлом.

```
bool VS_Bind( PSTR szFileName );
```

Параметры:

szFileName имя файла, содержащего первый кадр последовательности изображений, или имя AVI файла.

Возвращаемое значение: в случае успеха функция возвращает true, в случае неудачи - false.

Описание: Источником является последовательность BMP файлов или AVI файл. Имена файлов с изображениями (BMP) должны формироваться из префикса (необязательно) и числа. Все числа должны последовательно возрастать и должны состоять из одинакового количества цифр. Максимальное количество цифр в числе - 6. Например, последовательность может выглядеть следующим образом: Img0000.bmp, Img0001.bmp, Img0002.bmp и так далее. Также, в качестве источника можно задать имя файла, не содержащего числа. В этом случае всегда будет отображаться только один, заданный кадр. Отображение кадров происходит циклически, то есть после считывания последнего кадра будет считан первый кадр. Если в параметре функции передать NULL, или если программа не найдёт файл с заданным именем, то функция покажет диалоговое окно, в котором можно выбрать первый файл последовательности изображений или AVI файл.

4.1.1.2. VS_CreatelImage

Создание окна для вывода изображения.

```
bool VS_CreatelImage( char *szName, int nID, int nWidth, int nHeight,
int nType, S_VS_Pal *pPalette );
```

Параметры:

szName название окна. Передаваемая строка будет отображена в заголовке окна.

nID идентификатор окна - уникальное целочисленное значение. Все дальнейшие обращения к окну и к изображению, связанному с этим окном будут происходить через заданный идентификатор. Если уже создано окно с таким идентификатором, то функция возвратит false и окно не будет создано.

nWidth ширина клиентской области окна, или ширина выводимого в данное окно изображения. Ширина задаётся в пикселях.

nHeight высота (количество строк) клиентской области окна, или высота выводимого в данное окно изображения. Высота задаётся в пикселях.

nType [тип выводимого изображения](#) - одна из констант, задающая тип изображения. Тип определяет цветовую модель изображения.

pPalette палитра. Указатель на массив структур [S_VS_Pal](#). Значение этого параметра игнорируется для непалитровых изображений (см. описание типов изображений). Если этот параметр установлен в NULL для палитровых изображений, то будет использоваться полутоновая (градации серого) палитра.

Возвращаемое значение: в случае успеха функция возвращает true, в случае неудачи - false.

Описание: Функция создаёт дочернее MDI окно для вывода изображения размером *nWidth* на *nHeight* пикселей и цветовой организацией, заданной параметром *nType* и палитрой.

4.1.1.3. VS_DeletelImage

Удаление раннее созданного окна для вывода изображений.

```
bool VS_DeletelImage( int nID );
```

Параметры:

nID идентификатор удаляемого окна.

Возвращаемое значение: в случае успеха функция возвращает true, в случае неудачи - false.

Описание: Функция удаляет раннее созданное окно для вывода изображений с заданным идентификатором и освобождает все связанные с окном ресурсы. Идентификатор удалённого окна можно использовать повторно при создании нового окна.

4.1.1.4. VS_Draw

Вывод изображения

```
bool VS_Draw( int nID );
```

Параметры:

nID идентификатор окна.

Возвращаемое значение: в случае успеха функция возвращает true, в случае неудачи - false.

Описание: Функция выводит ранее установленные функцией [VS_SetData\(\)](#) данные в заданное окно. Если в параметре задан идентификатор [VS_DRAW_ALL](#), то будут отрисованы изображения во всех созданных окнах.

4.1.1.5. VS_GetMouseStatus

Получение состояния мыши

```
bool VS_GetMouseStatus( S_VSMouseStatus *pMouseStatus );
```

Параметры:

pMouseStatus структура с описанием состояния мыши.

Возвращаемое значение: в случае успеха функция возвращает true, в случае неудачи - false.

Описание: Функция заполняет структуру с описанием состояния мыши [S_VS_MouseStatus](#) на момент вызова функции VS_GetMouseStatus().

4.1.1.6. VS_GetSrcFrameNum

Получение текущего номера кадра источника.

```
int VS_GetSrcFrameNum();
```

Параметры: нет.

Возвращаемое значение: текущий номер кадра источника, начиная с 1.

4.1.1.7. VS_GetSrcFrames

Получение количества кадров в AVI файле, или в последовательности изображений

```
int VS_GetSrcFrames();
```

Параметры: нет.

Возвращаемое значение: количество кадров в источнике изображений (в AVI файле, или в последовательности изображений).

4.1.1.8. VS_Init

Инициализация библиотеки.

```
bool VS_Init();
```

Параметры: нет.

Возвращаемое значение: в случае успеха функция возвращает true, в случае неудачи - false.

Описание: Функцию инициализации необходимо вызвать до начала вызовов других функций библиотеки.


4.1.1.9. VS_OpPause

.

```
bool VS_OpPause();
```

Параметры: нет.

Возвращаемое значение: в случае успеха функция возвращает true, в случае неудачи - false.

Описание: Функция производит действие, эквивалентное нажатию кнопки  на панели инструментов.


4.1.1.10. VS_OpReset

.

```
bool VS_OpReset();
```

Параметры: нет.

Возвращаемое значение: в случае успеха функция возвращает true, в случае неудачи - false.

Описание: Функция производит действие, эквивалентное нажатию кнопки  на панели инструментов.


4.1.1.11. VS_OpRunBackward

.

```
bool VS_OpRunBackward();
```

Параметры: нет.

Возвращаемое значение: в случае успеха функция возвращает true, в случае неудачи - false.

Описание: Функция производит действие, эквивалентное нажатию кнопки  на панели инструментов.


4.1.1.12. VS_OpRunForward

.

```
bool VS_OpRunForward();
```

Параметры: нет.

Возвращаемое значение: в случае успеха функция возвращает true, в случае неудачи - false.

Описание: Функция производит действие, эквивалентное нажатию кнопки  на панели инструментов.


4.1.1.13. VS_OpStepBackward

.

```
bool VS_OpStepBackward();
```

Параметры: нет.

Возвращаемое значение: в случае успеха функция возвращает true, в случае неудачи - false.

Описание: Функция производит действие, эквивалентное нажатию кнопки  на панели инструментов.


4.1.1.14. VS_OpStepForward

.

```
bool VS_OpStepForward();
```

Параметры: нет.

Возвращаемое значение: в случае успеха функция возвращает true, в случае неудачи - false.

Описание: Функция производит действие, эквивалентное нажатию кнопки  на панели инструментов.

4.1.1.15. VS_Release

Деинициализация библиотеки.

```
void VS_Release();
```

Параметры: нет.

Возвращаемое значение: нет.

Описание: Функция закрывает окно приложения и освобождает занимаемые ресурсы. Эту функцию можно не вызывать, если окно приложения было закрыто пользователем.

4.1.1.16. VS_Run

Функция для организации основного программного цикла.

```
int VS_Run();
```

Параметры: нет.

Возвращаемое значение: Функция возвращает комбинацию [флагов состояния](#) (логическое "или").

Описание: В основном программном цикле происходит считывание очередного кадра из источника (если источник был задан, то есть если была вызвана функция [VS_Bind\(\)](#)) и отображение изображений. Функция блокирует работу программы и не возвращает управление до тех пор, пока не произойдёт одно из следующих событий:

Пользователь начал последовательный вывод видеокадров.

Пользователь начал пошаговый вывод видеокадров.

Пользователь закрыл программную "оболочку".

4.1.1.17. VS_Seek

Переход к заданному кадру

```
bool VS_Seek( int nFrameNum );
```

Параметры:

nFrameNum номер кадра.

Возвращаемое значение: нет.

Описание: Функция устанавливает номер кадра с которого будет начинаться новая последовательность кадров. Если *nFrameNum* меньше номера первого кадра, то новая последовательность будет начинаться с первого кадра. Если *nFrameNum* больше номера последнего кадра, то новая последовательность будет начинаться с последнего кадра.

4.1.1.18. VS_SetVectors

Установка векторов движения

```
bool VS_SetVectors( int nID, int nBlockSize, S_VS_Point *pVectors,
unsigned int unVectorsColor );
```

Параметры:

nID идентификатор окна.

nBlockSize размер блока NxN в пикселях.

pVectors буфер с данными векторов движения.

unVectorsColor цвет для отрисовки векторов движения.

Возвращаемое значение: в случае успеха функция возвращает true, в случае неудачи - false.

Описание: Функция передаёт данные о векторах движения, которые могут быть наложены на ассоциированное с окном с идентификатором *nID* изображение. Количество

передаваемых векторов зависит от размера изображения и размера блока `nBlockSize`. Количество векторов можно определить по формуле: $(\text{ширина} * \text{высота}) / (\text{nBlockSize} * \text{nBlockSize})$. Данные о каждом векторе представлены в структуре [S_VS_Point](#). Бывают случаи, когда требуется пометить вектор как "несуществующий" - признак того, что данный блок не имеет вектора движения. Следует отличать такие векторы от нулевых векторов. Нулевые векторы всё же помечаются точкой цветом, который получается инверсией цвета векторов, а отсутствующие векторы никак не помечаются. Пометить вектор как несуществующий можно, установив в компоненте `nX` значение [VS_NULL_VEC](#). В функцию можно передать значение цвета для отображения векторов. Цвет вектора задаётся в 32-х битовом слове. Цвет в этом слове определяется следующим образом: в первом (младшем) байте содержится интенсивность красного цвета, во втором байте содержится интенсивность зелёного цвета, в третьем байте содержится интенсивность голубого цвета, четвёртый байт не используется. Для формирования этого слова можно воспользоваться макросом `RGB()`, определённым в `wingdi.h`. Если параметр `unVectorsColor` установить в [VS_DEF_VEC_COLOR](#), то векторы будут отображаться цветом по умолчанию, который задаётся пользователем из приложения.

4.1.1.19. VS_SetVectorsF

Установка векторов движения

```
bool VS_SetVectorsF( int nID, int nBlockSize, S_VS_PointF *pVectors,
unsigned int unVectorsColor );
```

Параметры:

nID идентификатор окна.

nBlockSize размер блока NxN в пикселях.

pVectors буфер с данными векторов движения.

unVectorsColor цвет для отрисовки векторов движения.

Возвращаемое значение: в случае успеха функция возвращает `true`, в случае неудачи - `false`.

Описание: Функция передаёт данные о векторах движения, которые могут быть наложены на ассоциированное с окном с идентификатором `nID` изображение. Количество передаваемых векторов зависит от размера изображения и размера блока `nBlockSize`. Количество векторов можно определить по формуле: $(\text{ширина} * \text{высота}) / (\text{nBlockSize} * \text{nBlockSize})$. Данные о каждом векторе представлены в структуре [S_VS_PointF](#). Бывают случаи, когда требуется пометить вектор как "несуществующий" - признак того, что данный блок не имеет вектора движения. Следует отличать такие векторы от нулевых векторов. Нулевые векторы всё же помечаются точкой цветом, который получается инверсией цвета векторов, а отсутствующие векторы никак не помечаются. Пометить вектор как несуществующий можно, установив в компоненте `fltX` значение [\(float\)VS_NULL_VEC](#). В функцию можно передать значение цвета для отображения векторов. Цвет вектора задаётся в 32-х битовом слове. Цвет в этом слове определяется следующим образом: в первом (младшем) байте содержится интенсивность красного цвета, во втором байте содержится интенсивность зелёного цвета, в третьем байте содержится интенсивность голубого цвета, четвёртый байт не используется. Для

формирования этого слова можно воспользоваться макросом RGB(), определённым в wingdi.h. Если параметр unVectorsColor установить в [VS_DEF_VEC_COLOR](#), то векторы будут отображаться цветом по умолчанию, который задаётся пользователем из приложения.

4.1.1.20. VS_Text

Отображение текста

```
bool VS_Text( const char *szText, ... );
```

Параметры:

szText строка со спецификаторами форматирования.
... аргументы.

Возвращаемое значение: в случае успеха функция возвращает true, в случае неудачи - false.

Описание: Функция формирует строку с учётом спецификаторов форматирования и аргументов и выводит её в специальное текстовое MDI окно. Информацию о спецификаторах форматирования можно найти в описании функции printf() стандартной библиотеки ввода-вывода C. Функция VS_Text формирует строку также, как функция printf(). Специальное текстовое MDI окно создаётся и отображается, после первого вызова функции VS_Text(). Если VS_Text() не вызывается в пользовательской программе, то текстовое окно не создаётся. Отображаемый в окне текст можно просматривать с помощью прокрутки. Размер буфера для вывода текста ограничен, поэтому при внесении новой записи используется циклический буфер, то есть при внесении новой записи в заполненный буфер уничтожается самая старая запись. Вывод в текстовое окно можно рассматривать как альтернативу вывода сообщений на консольное окно.

4.1.2. Функции получения и установки данных изображения

4.1.2.1. VS_GetBGRData

Получение цветных данных изображения

```
bool VS_GetBGRData( int nID, void *pvData );
```

Параметры:

nID идентификатор окна.
pvData буфер для получаемых данных.

Возвращаемое значение: в случае успеха функция возвращает true, в случае неудачи - false.

Описание: Функция получает визуальные данные от дочернего MDI окна с идентификатором, заданным параметром nID. Если в качестве идентификатора передано

значение [VS_SOURCE](#), то будут получены данные от источника видеокадров. Получаемые данные всегда имеют тип [VS_RGB24](#) и размер, соответствующий размерам окна не зависимо от того какую цветовую организацию имеет окно. Следует следить за тем, чтобы буфер pvData имел размер, достаточный для размещения получаемых данных. Получить необходимый размер для буфера pvData можно с помощью вызовов [VS_GetWidth\(\)](#) и [VS_GetHeight\(\)](#). Требуемый размер буфера - это утроенное произведение ширины и высоты окна.

4.1.2.2. VS_GetData

Получение данных изображения

```
bool VS_GetData( int nID, void *pvData );
```

Параметры:

nID идентификатор окна.

pvData буфер для получаемых данных.

Возвращаемое значение: в случае успеха функция возвращает true, в случае неудачи - false.

Описание: Функция получает визуальные данные от дочернего MDI окна с идентификатором, заданным параметром nID. Если в качестве идентификатора передано значение [VS_SOURCE](#), то будут получены данные от источника видеокадров. Структура и размер (ширина, высота и цветовая организация изображения) получаемых данных соответствуют типу и размеру окна. Следует следить за тем, чтобы буфер pvData имел размер, достаточный для размещения получаемых данных. Получить необходимый размер для буфера pvData можно с помощью вызова [VS_GetBufSize\(\)](#).

4.1.2.3. VS_GetGrayData

Получение полутоновых (256 градаций серого цвета) данных изображения

```
bool VS_GetGrayData( int nID, void *pvData );
```

Параметры:

nID идентификатор окна.

pvData буфер для получаемых данных.

Возвращаемое значение: в случае успеха функция возвращает true, в случае неудачи - false.

Описание: Функция получает визуальные данные от дочернего MDI окна с идентификатором, заданным параметром nID. Если в качестве идентификатора передано значение [VS_SOURCE](#), то будут получены данные от источника видеокадров. Получаемые данные всегда имеют тип [VS_RGB8](#) с полутоновой палитрой и размер, соответствующий размерам окна не зависимо от того какую цветовую организацию имеет окно. Следует следить за тем, чтобы буфер pvData имел размер, достаточный

для размещения получаемых данных. Получить необходимый размер для буфера `pvData` можно с помощью вызовов [VS_GetWidth\(\)](#) и [VS_GetHeight\(\)](#). Требуемый размер буфера - это произведение ширины и высоты окна.

4.1.2.4. VS_SetData

Установка данных изображения

```
bool VS_SetData( int nID, void *pvData );
```

Параметры:

nID идентификатор окна.

pvData буфер передаваемых данных.

Возвращаемое значение: в случае успеха функция возвращает `true`, в случае неудачи - `false`.

Описание: Функция передаёт визуальные данные дочернему MDI окну с идентификатором, заданным параметром `nID`. Структура и размер передаваемых данных (ширина, высота и цветовая организация изображения) должны соответствовать типу и размеру окна. Если в первом параметре используется идентификатор [VS_SOURCE](#), то функция вернёт `false`, так как невозможно установить данные в источник изображений.

4.1.3. Функции получения параметров изображения

4.1.3.1. VS_BitsPerPixel

Получение количества битов на пиксель изображения

```
int VS_GetBitsPerPixel( int nID );
```

Параметры:

nID идентификатор окна.

Возвращаемое значение: количество битов на пиксель ассоциированного с окном с идентификатором `nID` изображения.

Описание: Если в качестве идентификатора окна передано значение [VS_SOURCE](#), то будет возвращено количество битов на пиксель изображения источника видеокадров.

4.1.3.2. VS_GetBufSize

Получение размера буфера, необходимого для размещения изображения

```
int VS_GetBufSize( int nID );
```

Параметры:

nID идентификатор окна.

Возвращаемое значение: размер буфера, необходимого для размещения ассоциированного с окном с идентификатором `nID` изображения.

Описание: Размер буфера возвращается в байтах. Если в качестве идентификатора окна передано значение [VS_SOURCE](#), то будет возвращён размер буфера изображения источника видеок кадров.

4.1.3.3. VS_GetHeight

Получение высоты изображения

```
int VS_GetHeight( int nID );
```

Параметры:

`nID` идентификатор окна.

Возвращаемое значение: высота ассоциированного с окном с идентификатором `nID` изображения в пикселях.

Описание: Если в качестве идентификатора окна передано значение [VS_SOURCE](#), то будет возвращена высота изображения источника видеок кадров.

4.1.3.4. VS_GetType

Получение идентификатора типа изображения

```
int VS_GetType( int nID );
```

Параметры:

`nID` идентификатор окна.

Возвращаемое значение: [идентификатор типа](#) ассоциированного с окном с идентификатором `nID` изображения в пикселях.

Описание: Если в качестве идентификатора окна передано значение [VS_SOURCE](#), то будет возвращён [идентификатор типа](#) источника видеок кадров.

4.1.3.5. VS_GetWidth

Получение ширины изображения

```
int VS_GetWidth( int nID );
```

Параметры:

`nID` идентификатор окна.

Возвращаемое значение: ширина ассоциированного с окном с идентификатором `nID` изображения в пикселях.

Описание: Если в качестве идентификатора окна передано значение [VS_SOURCE](#), то будет возвращена ширина изображения источника видеок кадров.

4.1.4. Функции для работы с палитрами

4.1.4.1. VS_CreateCustomPalette

Функция создает кусочно-линейную палитру `pPalette`, интерполированную по массиву опорных точек

```
void VS_CreateCustomPalette( S\_VS\_Pal *pPalette, VS\_STOP\_COLOR
*pStopColor, int count );
```

Параметры:

pPalette генерируемая палитра.
pStopColor массив опорных цветов.
count длина массива `pStopColor`.

Возвращаемое значение: нет.

Описание: значения палитры в диапазоне от 0 до `pStopColor[0].index`, заполняется значением `pStopColor[0].rgb`. Значения палитры в диапазоне от `pStopColor[count-1].index` до 255, заполняется значением `pStopColor[count-1].rgb`.

4.1.4.2. VS_GetPalette

Получение палитры

```
bool VS_GetPalette( int nID, S\_VS\_Pal *pPal );
```

Параметры:

nID идентификатор окна.
pPal буфер для получаемой палитры.

Возвращаемое значение: в случае успеха функция возвращает `true`, в случае неудачи - `false`.

Описание: Функция получает палитру изображения, ассоциированного с окном с идентификатором `nID`. Получаемая палитра - это массив элементов [S_VS_Pal](#). Количество элементов в массиве определяется цветовой организацией изображения. Для 1 битовых изображений требуется 2 элемента, для 4-х битовых изображений требуется 16 элементов, для 8-ми битовых изображений требуется 256 элементов. При получении палитры для не палитровых изображений функция возвращает `false`.

4.1.4.3. VS_SetCirclePalette

Установка круговой палитры

```
bool VS_SetCirclePalette( int nID, int nColorDisp );
```

Параметры:

nID идентификатор окна.

nColorDisp параметр, задающий поворот цветового круга. При *nColorDisp*=0 жёлтый цвет находится сверху, а синий снизу.

Возвращаемое значение: в случае успеха функция возвращает true, в случае неудачи - false.

Описание: Функция устанавливает палитру для изображения, ассоциированного с окном с идентификатором *nID*. Устанавливаемая палитра - это палитра с плавно меняющимися цветами в следующем порядке: жёлтый - зелёный - циановый - голубой - пурпурный - красный, в зависимости от параметра *nColorDisp*. На рисунке показан цветовой круг и соответствующие ему значения при минимальном и максимальном значении пикселя (направление обхода - по часовой стрелке).

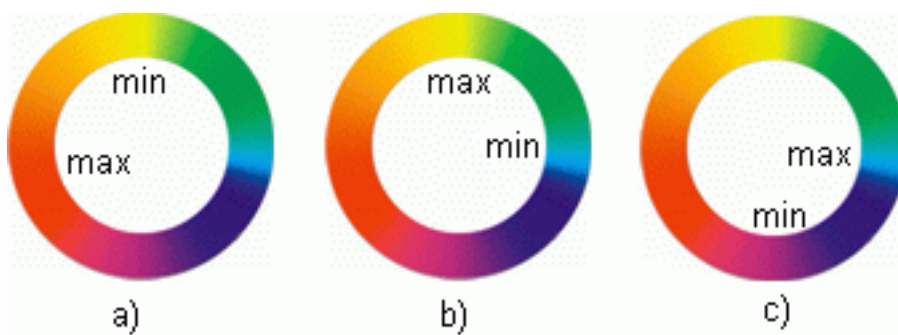


Рисунок 4.1 -

a) при *nColorDisp* = 0.

b) при *nColorDisp* = 63.

c) при *nColorDisp* = 127.

При установке палитры для не палитровых изображений функция не производит никаких действий и возвращает true.

4.1.4.4. VS_SetGrayPalette

Установка полутоновой (серой) палитры

```
bool VS_SetGrayPalette( int nID );
```

Параметры:

nID идентификатор окна.

Возвращаемое значение: в случае успеха функция возвращает true, в случае неудачи - false.

Описание: Функция устанавливает палитру для изображения, ассоциированного с окном с идентификатором *nID*. Устанавливаемая палитра заполняется градациями серого цвета

- от чёрного к белому. При установке палитры для не палитровых изображений функция не производит никаких действий и возвращает true.

4.1.4.5. VS_SetPalette

Установка палитры

```
bool VS_SetPalette( int nID, S_VS_Pal *pPal );
```

Параметры:

nID идентификатор окна.

pPal устанавливаемая палитра.

Возвращаемое значение: в случае успеха функция возвращает true, в случае неудачи - false.

Описание: Функция устанавливает палитру для изображения, ассоциированного с окном с идентификатором nID. Устанавливаемая палитра - это массив элементов [S_VS_Pal](#). Количество элементов в массиве определяется цветовой организацией изображения. Для 1 битовых изображений требуется 2 элемента, для 4-х битовых изображений требуется 16 элементов, для 8-ми битовых изображений требуется 256 элементов. При установке палитры для не палитровых изображений функция не производит никаких действий и возвращает true.

4.1.4.6. VS_SetRandomPalette

Установка случайной палитры

```
bool VS_SetRandomPalette( int nID );
```

Параметры:

nID идентификатор окна.

Возвращаемое значение: в случае успеха функция возвращает true, в случае неудачи - false.

Описание: Функция устанавливает палитру для изображения, ассоциированного с окном с идентификатором nID. Устанавливаемая палитра заполняется случайными значениями. При установке палитры для не палитровых изображений функция не производит никаких действий и возвращает true.

4.1.5. Функции для работы с элементами управления

4.1.5.1. VS_CheckRadio

Включение переключателя

```
bool VS_CheckRadio( int nGroupID, int nRadioID, bool fCheck );
```

Параметры:

nGroupID идентификатор группы переключателей.

nRadioID идентификатор переключателя.

fCheck флаг включения переключателя.

Возвращаемое значение: в случае успеха функция возвращает true, в случае неудачи - false.

Описание: Если *fCheck* равно true, то функция включает переключатель с идентификатором *nRadioID* из группы с идентификатором *nGroupID*, а все остальные переключатели из этой группы выключает. Если *fCheck* равно false, то функция выключает переключатель с идентификатором *nRadioID* из группы с идентификатором *nGroupID*.

4.1.5.2. VS_CreateCheckBox

Создаёт выключатель.

```
bool VS_CreateCheckBox( char *szName, int nID, bool fCheck );
```

Параметры:

szName имя выключателя.

nID идентификатор выключателя.

fCheck начальное состояние выключателя. true - включен, false - выключен.

Возвращаемое значение: в случае успеха функция возвращает true, в случае неудачи - false.

Описание: Выключатель - это элемент управления, который может находиться в одном из двух состояний - включено или выключено. Функция создаёт этот элемент управления с именем *szName* и идентификатором *nID*. После создания выключателя VShell устанавливает его в состояние *fCheck*. Созданный выключатель размещается в специальном окне для элементов управления. Если при создании выключателя указан идентификатор уже существующего выключателя, то новый элемент управления не будет создан и функция возвратит false.

4.1.5.3. VS_CreateEdit

Создание поля ввода

```
bool VS_CreateEdit( char *szName, int nID );
```

Параметры:

szName имя поля ввода

nID идентификатор поля ввода

Возвращаемое значение: в случае успеха функция возвращает true, в случае неудачи - false.

Описание: Поле ввода - это элемент управления, который позволяет ввести некоторое значение - текстовую строку, или число. После ввода данных необходимо нажать клавишу <ENTER>, показав, таким образом, что ввод значения закончен. Созданное поле ввода размещается в специальном окне для элементов управления. Если при создании поля ввода указан идентификатор уже существующего поля ввода, то новый элемент управления не будет создан и функция возвратит false.

4.1.5.4. VS_CreateRadioGroup

Создание группы переключателей (радио кнопки)

```
bool VS_CreateRadioGroup( char *szGroupName, int nGroupID, int
nQuantity, char *szRadioName, int *pnRadioID, int nChecked );
```

Параметры:

szGroupName имя группы переключателей
nGroupID идентификатор группы переключателей
nQuantity количество переключателей в группе
szRadioName имена переключателей
pnRadioID массив с идентификаторами переключателей
nChecked порядковый номер (начиная с нуля) переключателя, который будет включён после создания группы переключателей.

Возвращаемое значение: в случае успеха функция возвращает true, в случае неудачи - false.

Описание: Переключатели - это элемент управления, который находится в одном из двух состояний: включен и выключен. Причём, если эти элементы объединены в группу, то при включении одного элемента группы автоматически выключаются все остальные элементы группы. Функция создаёт группу таких переключателей с именем *szGroupName* и идентификатором *nGroupID*. Количество переключателей в группе определяется параметром *nQuantity*. Каждому переключателю в группе нужно присвоить имя и уникальный в пределах группы идентификатор. Имена для всех переключателей передаются в одной строке *szRadioName*. Имена в строке задаются последовательно для каждого из переключателей и разделяются символом '&'. Например, если в группе три переключателя, то строка *szRadioName* может выглядеть так: "Radio1&Radio2&Radio3", где "Radio1" - имя первого переключателя, "Radio2" - имя второго переключателя, а "Radio3" - имя третьего переключателя. В массиве *pnRadioID* последовательно задаются идентификаторы для каждого из переключателей. После создания группы переключателей VShell включит переключатель с порядковым номером (начиная с нуля) *nChecked*. Если *nChecked* больше или равно *nQuantity*, то все переключатели будут выключены. Созданная группа переключателей размещается в специальном окне для элементов управления. Если при создании группы переключателей указан идентификатор уже существующей группы, то новая группа не будет создана и функция возвратит false.

4.1.5.5. VS_CreateSlider

Создание элемента управления "Слайдер"

```
bool VS_CreateSlider( char *szName, int nID, float fltMin, float fltMax,
float fltStep, float fltStartPos );
```

Параметры:

szName имя слайдера.

nID идентификатор слайдера.

fltMin минимальное значение рабочего диапазона слайдера.

fltMax максимальное значение рабочего диапазона слайдера.

fltStep шаг бегунка слайдера

fltStartPos начальная позиция

Возвращаемое значение: в случае успеха функция возвращает true, в случае неудачи - false.

Описание: Слайдер - это элемент управления с бегунком, с помощью которого можно выбрать значения от *fltMin* до *fltMax* включительно с шагом *fltStep*. После создания бегунок слайдера устанавливается в начальную позицию *fltStartPos*. Если начальная позиция меньше *fltMin*, то слайдер устанавливается в позицию *fltMin*. Если начальная позиция больше *fltMax*, то слайдер устанавливается в позицию *fltMax*. Созданный слайдер размещается в специальном окне для элементов управления. Если при создании слайдера указан идентификатор уже существующего слайдера, то новый слайдер не будет создан и функция возвратит false.

4.1.5.6. VS_GetCheckBox

Получение состояния выключателя

```
bool VS_GetCheckBox( int nID );
```

Параметры:

nID идентификатор выключателя

Возвращаемое значение: Если выключатель включен, то функция возвращает true, если выключен, то функция возвращает false.

4.1.5.7. VS_GetCheckedID

Получение идентификатора включенного переключателя

```
int VS_GetCheckedID( int nGroupID );
```

Параметры:

nGroupID идентификатор группы радио кнопок.

Возвращаемое значение: возвращает идентификатор включенного на момент вызова функции переключателя из группы с идентификатором *nGroupID*. Если все переключатели выключены, то функция возвращает 0.

4.1.5.8. VS_GetEditFloat

Получение числа с плавающей точкой из поля ввода

```
float VS_GetEditFloat( int nID, bool *pfError );
```

Параметры:

nID идентификатор поля ввода

pfError флаг ошибки

Возвращаемое значение: в случае успеха функция возвращает true, в случае неудачи - false.

Описание: Если из поля ввода не возможно получить число с плавающей точкой, то флаг ошибки будет равен true. Если не требуется анализировать ошибку, то вместо параметра pfError можно передать NULL.

4.1.5.9. VS_GetEditInt

Получение целочисленного значения из поля ввода

```
int VS_GetEditInt( int nID, bool *pfError );
```

Параметры:

nID идентификатор поля ввода

pfError флаг ошибки.

Возвращаемое значение: в случае успеха функция возвращает true, в случае неудачи - false.

Описание: Если из поля ввода не возможно получить целочисленное значение, то флаг ошибки будет равен true. Если не требуется анализировать ошибку, то вместо параметра pfError можно передать NULL.

4.1.5.10. VS_GetEditText

Получение текстовой строки из поля ввода

```
bool VS_GetEditText( int nID, PSTR szText );
```

Параметры:

nID идентификатор поля ввода

szText буфер для считываемой строки

Возвращаемое значение: в случае успеха функция возвращает true, в случае неудачи - false.

4.1.5.11. VS_GetSlider

Получение текущей позиции слайдера

```
float VS_GetSlider( int nID );
```

Параметры:

nID идентификатор слайдера.

Возвращаемое значение: позиция слайдера.

Описание: Функция возвращает текущую позицию слайдера с идентификатором *nID*.

4.1.5.12. VS_IsRadioChecked

Получение состояния переключателя

```
bool VS_IsRadioChecked( int nGroupID, int nRadioID );
```

Параметры:

nGroupID идентификатор группы переключателей.

nRadioID идентификатор переключателя.

Возвращаемое значение: функция возвращает true, если включен переключатель с идентификатором *nRadioID* из группы с идентификатором *nGroupID*. Если переключатель выключен, то функция возвращает false.

4.1.5.13. VS_IsSliderChanged

Флаг изменения позиции слайдера

```
bool VS_IsSliderChanged( int nID );
```

Параметры:

nID идентификатор слайдера.

Возвращаемое значение: Если позиция слайдера изменилась после последнего вызова [VS_GetSlider\(\)](#), то функция возвращает true, если позиция не изменилась, то функция возвращает false.

Описание: Функция возвращает флаг изменения позиции слайдера с идентификатором *nID*.

4.1.5.14. VS_SetCheckBox

Установка состояния выключателя

```
bool VS_SetCheckBox( int nID, bool fCheck );
```

Параметры:

nID идентификатор выключателя

fCheck новое состояние выключателя true - включен, false - выключен.

Возвращаемое значение: в случае успеха функция возвращает true, в случае неудачи - false.

4.1.5.15. VS_SetEditFloat

Установка числа с плавающей точкой в поле ввода

```
bool VS_SetEditFloat( int nID, float fltVal );
```

Параметры:

nID идентификатор поля ввода

fltVal устанавливаемое значения

Возвращаемое значение: в случае успеха функция возвращает true, в случае неудачи - false.

4.1.5.16. VS_SetEditInt

Установка целого значения в поле ввода

```
bool VS_SetEditInt( int nID, int nVal );
```

Параметры:

nID идентификатор поля ввода

nVal устанавливаемое значение

Возвращаемое значение: в случае успеха функция возвращает true, в случае неудачи - false.

4.1.5.17. VS_SetEditText

Установка текстовой строки в поле ввода

```
bool VS_SetEditText( int nID, char *szText );
```

Параметры:

nID идентификатор поля ввода.

szText устанавливаемая строка.

Возвращаемое значение: в случае успеха функция возвращает true, в случае неудачи - false.

4.1.5.18. VS_SetSlider

Установка новой позиции слайдера

```
bool VS_SetSlider( int nID, float fltVal );
```

Параметры:

nID идентификатор окна.

fltVal новое значение позиции слайдера.

Возвращаемое значение: в случае успеха функция возвращает true, в случае неудачи - false.

Описание: Функция устанавливает слайдер с идентификатором *nID* в позицию *fltVal*. Если *fltVal* меньше минимального значения рабочего диапазона слайдера, то слайдер устанавливается в минимальную позицию диапазона. Если *fltVal* больше максимального значения рабочего диапазона слайдера, то слайдер устанавливается в максимальную позицию диапазона.

4.1.6. Функции векторной графики

4.1.6.1. VS_Arc

Отрисовка дуги

```
bool VS_Arc( int nID, int nLeftRect, int nTopRect, int nRightRect, int
nBottomRect, int nXStartArc, int nYStartArc, int nXEndArc, int nYEndArc,
unsigned int unColor );
```

Параметры:

nID идентификатор окна.

nLeftRect X координата левого верхнего угла прямоугольника, описанного около эллипса, на котором лежит дуга.

nTopRect Y координата левого верхнего угла прямоугольника, описанного около эллипса, на котором лежит дуга.

nRightRect X координата правого нижнего угла прямоугольника, описанного около эллипса, на котором лежит дуга.

nBottomRect Y координата правого нижнего угла прямоугольника, описанного около эллипса, на котором лежит дуга.

nXStartArc X координата точки на начальном радиусе дуги.

nYStartArc Y координата точки на начальном радиусе дуги.

nXEndArc X координата точки на конечном радиусе дуги.

nYEndArc Y координата точки на конечном радиусе дуги.

unColor цвет дуги.

Возвращаемое значение: в случае успеха функция возвращает true, в случае неудачи - false.

Описание: Дуга является частью линии, ограничивающей эллипс. При вызове функции необходимо задать эллипс и указать ту его часть, которая должна содержаться в нарисованной дуге. Для задания эллипса необходимо указать координаты

прямоугольника, который описывается около эллипса с помощью параметров `nLeftRect`, `nTopRect`, `nRightRect` и `nBottomRect`. Функция рисует дугу от той точки, где начальный радиус пересекается с эллипсом, до точки его пересечения с конечным радиусом в направлении против часовой стрелки. Начальный радиус - это прямая, проходящая через центр эллипса и точку с координатами [`nXStartArc`, `nYStartArc`]. Конечный радиус - это прямая, проходящая через центр эллипса и точку с координатами [`nXEndArc`, `nYEndArc`]. Цвет дуги задаётся так, как описано в функции [VS_SetPixel\(\)](#).

4.1.6.2. VS_ArcEx

Отрисовка дуги с возможностью установки стиля пера

```
bool VS_ArcEx( int nID, int nLeftRect, int nTopRect, int nRightRect, int
nBottomRect, int nXStartArc, int nYStartArc, int nXEndArc, int nYEndArc,
unsigned int unColor, int nDashDot );
```

Параметры:

nID идентификатор окна.

nLeftRect X координата левого верхнего угла прямоугольника, описанного около эллипса, на котором лежит дуга.

nTopRect Y координата левого верхнего угла прямоугольника, описанного около эллипса, на котором лежит дуга.

nRightRect X координата правого нижнего угла прямоугольника, описанного около эллипса, на котором лежит дуга.

nBottomRect Y координата правого нижнего угла прямоугольника, описанного около эллипса, на котором лежит дуга.

nXStartArc X координата точки на начальном радиусе дуги.

nYStartArc Y координата точки на начальном радиусе дуги.

nXEndArc X координата точки на конечном радиусе дуги.

nYEndArc Y координата точки на конечном радиусе дуги.

unColor цвет дуги.

nDashDot стиль пера, используемый при отрисовки линий. Для задания стиля используются [идентификаторы стиля пера](#), или 0 - если требуется отрисовка сплошной линией.

Возвращаемое значение: в случае успеха функция возвращает true, в случае неудачи - false.

Описание: Дуга является частью линии, ограничивающей эллипс. При вызове функции необходимо задать эллипс и указать ту его часть, которая должна содержаться в нарисованной дуге. Для задания эллипса необходимо указать координаты прямоугольника, который описывается около эллипса с помощью параметров `nLeftRect`, `nTopRect`, `nRightRect` и `nBottomRect`. Функция рисует дугу от той точки, где начальный радиус пересекается с эллипсом, до точки его пересечения с конечным радиусом в направлении против часовой стрелки. Начальный радиус - это прямая, проходящая через центр эллипса и точку с координатами [`nXStartArc`, `nYStartArc`]. Конечный радиус - это прямая, проходящая через центр эллипса и точку с координатами [`nXEndArc`, `nYEndArc`]. Цвет дуги задаётся так, как описано в функции [VS_SetPixel\(\)](#).

4.1.6.3. VS_ArcF

Отрисовка дуги

```
bool VS_ArcF( int nID, float fltLeftRect, float fltTopRect, float fltRightRect,
float fltBottomRect, float fltXStartArc, float fltYStartArc, float fltXEndArc,
float fltYEndArc, unsigned int unColor );
```

Параметры:

nID идентификатор окна.

fltLeftRect X координата левого верхнего угла прямоугольника, описанного около эллипса, на котором лежит дуга.

fltTopRect Y координата левого верхнего угла прямоугольника, описанного около эллипса, на котором лежит дуга.

fltRightRect X координата правого нижнего угла прямоугольника, описанного около эллипса, на котором лежит дуга.

fltBottomRect Y координата правого нижнего угла прямоугольника, описанного около эллипса, на котором лежит дуга.

fltXStartArc X координата точки на начальном радиусе дуги.

fltYStartArc Y координата точки на начальном радиусе дуги.

fltXEndArc X координата точки на конечном радиусе дуги.

fltYEndArc Y координата точки на конечном радиусе дуги.

unColor цвет дуги.

Возвращаемое значение: в случае успеха функция возвращает true, в случае неудачи - false.

Описание: Версия функции [VS_Arc\(\)](#) для чисел с плавающей точкой.

4.1.6.4. VS_ArcFEx

Отрисовка дуги с возможностью установки стиля пера

```
bool VS_ArcFEx( int nID, float fltLeftRect, float fltTopRect, float
fltRightRect, float fltBottomRect, float fltXStartArc, float fltYStartArc, float
fltXEndArc, float fltYEndArc, unsigned int unColor, int nDashDot );
```

Параметры:

nID идентификатор окна.

fltLeftRect X координата левого верхнего угла прямоугольника, описанного около эллипса, на котором лежит дуга.

fltTopRect Y координата левого верхнего угла прямоугольника, описанного около эллипса, на котором лежит дуга.

fltRightRect X координата правого нижнего угла прямоугольника, описанного около эллипса, на котором лежит дуга.

fltBottomRect Y координата правого нижнего угла прямоугольника, описанного около эллипса, на котором лежит дуга.

fltXStartArc X координата точки на начальном радиусе дуги.

fltYStartArc Y координата точки на начальном радиусе дуги.

fltXEndArc X координата точки на конечном радиусе дуги.

fltYEndArc Y координата точки на конечном радиусе дуги.

unColor цвет дуги.

nDashDot стиль пера, используемый при отрисовки линий. Для задания стиля используются [идентификаторы стиля пера](#), или 0 - если требуется отрисовка сплошной линией.

Возвращаемое значение: в случае успеха функция возвращает true, в случае неудачи - false.

Описание: Версия функции [VS_ArcEx\(\)](#) для чисел с плавающей точкой.

4.1.6.5. VS_Chord

Отрисовка сегмента

```
bool VS_Chord( int nID, int nLeftRect, int nTopRect, int nRightRect, int
nBottomRect, int nXRadial1, int nYRadial1, int nXRadial2, int nYRadial2,
unsigned int unBoundColor, unsigned int unFillColor );
```

Параметры:

nID идентификатор окна.

nLeftRect X координата левого верхнего угла прямоугольника, описанного около эллипса, на котором лежит сегмент.

nTopRect Y координата левого верхнего угла прямоугольника, описанного около эллипса, на котором лежит сегмент.

nRightRect X координата правого нижнего угла прямоугольника, описанного около эллипса, на котором лежит сегмент.

nBottomRect Y координата правого нижнего угла прямоугольника, описанного около эллипса, на котором лежит сегмент.

nXRadial1 X координата точки на начальном радиусе сегмента.

nYRadial1 Y координата точки на начальном радиусе сегмента.

nXRadial2 X координата точки на конечном радиусе сегмента.

nYRadial2 Y координата точки на конечном радиусе сегмента.

unBoundColor цвет контура сегмента.

unFillColor цвет заполнителя сегмента.

Возвращаемое значение: в случае успеха функция возвращает true, в случае неудачи - false.

Описание: Сегмент является фигурой, образованной при пересечении эллипса с прямой (хорда). При вызове функции необходимо задать эллипс и указать ту его часть, которая должна содержаться в сегменте. Для задания эллипса необходимо указать координаты прямоугольника, который описывается около эллипса с помощью параметров *nLeftRect*, *nTopRect*, *nRightRect* и *nBottomRect*. Хорда - это прямая, пересекающая две точки. Первая точка хорды - точка пересечения начального радиуса с эллипсом. Вторая точка хорды - точка пересечения конечного радиуса с эллипсом. Начальный радиус - это прямая, проходящая через центр эллипса и точку с координатами [*nXRadial1*, *nYRadial1*].

Конечный радиус - это прямая, проходящая через центр эллипса и точку с координатами [nXRadial2, nYRadial2]. Функция рисует сегмент от первой точки хорды до второй по направлению против часовой стрелки. Цвет контура и цвет заполнения задаётся так, как описано в функции [VS_SetPixel\(\)](#). Если в параметре unFillColor установлено значение [VS_NULL_COLOR](#), то будет нарисован только контур сегмента, без заполнения его внутренней части.

4.1.6.6. VS_ChordEx

Отрисовка сегмента с возможностью установки стиля пера

```
bool VS_ChordEx( int nID, int nLeftRect, int nTopRect, int nRightRect,
int nBottomRect, int nXRadial1, int nYRadial1, int nXRadial2, int
nYRadial2, unsigned int unBoundColor, unsigned int unFillColor, int
nDashDot );
```

Параметры:

nID идентификатор окна.

nLeftRect X координата левого верхнего угла прямоугольника, описанного около эллипса, на котором лежит сегмент.

nTopRect Y координата левого верхнего угла прямоугольника, описанного около эллипса, на котором лежит сегмент.

nRightRect X координата правого нижнего угла прямоугольника, описанного около эллипса, на котором лежит сегмент.

nBottomRect Y координата правого нижнего угла прямоугольника, описанного около эллипса, на котором лежит сегмент.

nXRadial1 X координата точки на начальном радиусе сегмента.

nYRadial1 Y координата точки на начальном радиусе сегмента.

nXRadial2 X координата точки на конечном радиусе сегмента.

nYRadial2 Y координата точки на конечном радиусе сегмента.

unBoundColor цвет контура сегмента.

unFillColor цвет заполнителя сегмента.

nDashDot стиль пера, используемый при отрисовки контура. Для задания стиля используются [идентификаторы стиля пера](#), или 0 - если требуется отрисовка сплошной линией.

Возвращаемое значение: в случае успеха функция возвращает true, в случае неудачи - false.

Описание: Сегмент является фигурой, образованной при пересечении эллипса с прямой (хорда). При вызове функции необходимо задать эллипс и указать ту его часть, которая должна содержаться в сегменте. Для задания эллипса необходимо указать координаты прямоугольника, который описывается около эллипса с помощью параметров nLeftRect, nTopRect, nRightRect и nBottomRect. Хорда - это прямая, пересекающая две точки. Первая точка хорды - точка пересечения начального радиуса с эллипсом. Вторая точка хорды - точка пересечения конечного радиуса с эллипсом. Начальный радиус - это прямая, проходящая через центр эллипса и точку с координатами [nXRadial1, nYRadial1]. Конечный радиус - это прямая, проходящая через центр эллипса и точку с координатами

[nXRadial2, nYRadial2]. Функция рисует сегмент от первой точки хорды до второй по направлению против часовой стрелки. Цвет контура и цвет заполнения задаётся так, как описано в функции [VS_SetPixel\(\)](#). Если в параметре unFillColor установлено значение [VS_NULL_COLOR](#), то будет нарисован только контур сегмента, без заполнения его внутренней части.

4.1.6.7. VS_ChordF

Отрисовка сегмента

```
bool VS_ChordF( int nID, float fltLeftRect, float fltTopRect, float
fltRightRect, float fltBottomRect, float fltXRadial1, float fltYRadial1, float
fltXRadial2, float fltYRadial2, unsigned int unBoundColor, unsigned int
unFillColor );
```

Параметры:

nID идентификатор окна.

fltLeftRect X координата левого верхнего угла прямоугольника, описанного около эллипса, на котором лежит сегмент.

fltTopRect Y координата левого верхнего угла прямоугольника, описанного около эллипса, на котором лежит сегмент.

fltRightRect X координата правого нижнего угла прямоугольника, описанного около эллипса, на котором лежит сегмент.

fltBottomRect Y координата правого нижнего угла прямоугольника, описанного около эллипса, на котором лежит сегмент.

fltXRadial1 X координата точки на начальном радиусе сегмента.

fltYRadial1 Y координата точки на начальном радиусе сегмента.

fltXRadial2 X координата точки на конечном радиусе сегмента.

fltYRadial2 Y координата точки на конечном радиусе сегмента.

unBoundColor цвет контура сегмента.

unFillColor цвет заполнителя сегмента.

Возвращаемое значение: в случае успеха функция возвращает true, в случае неудачи - false.

Описание: Версия функции [VS_Chord\(\)](#) для чисел с плавающей точкой.

4.1.6.8. VS_ChordFEx

Отрисовка сегмента с возможностью установки стиля пера

```
bool VS_ChordFEx( int nID, float fltLeftRect, float fltTopRect, float
fltRightRect, float fltBottomRect, float fltXRadial1, float fltYRadial1, float
fltXRadial2, float fltYRadial2, unsigned int unBoundColor, unsigned int
unFillColor, int nDashDot );
```

Параметры:

nID идентификатор окна.

fltLeftRect X координата левого верхнего угла прямоугольника, описанного около эллипса, на котором лежит сегмент.

fltTopRect Y координата левого верхнего угла прямоугольника, описанного около эллипса, на котором лежит сегмент.

fltRightRect X координата правого нижнего угла прямоугольника, описанного около эллипса, на котором лежит сегмент.

fltBottomRect Y координата правого нижнего угла прямоугольника, описанного около эллипса, на котором лежит сегмент.

fltXRadial1 X координата точки на начальном радиусе сегмента.

fltYRadial1 Y координата точки на начальном радиусе сегмента.

fltXRadial2 X координата точки на конечном радиусе сегмента.

fltYRadial2 Y координата точки на конечном радиусе сегмента.

unBoundColor цвет контура сегмента.

unFillColor цвет заполнителя сегмента.

nDashDot стиль пера, используемый при отрисовки контура. Для задания стиля используются [идентификаторы стиля пера](#), или 0 - если требуется отрисовка сплошной линией.

Возвращаемое значение: в случае успеха функция возвращает true, в случае неудачи - false.

Описание: Версия функции [VS_ChordEx\(\)](#) для чисел с плавающей точкой.

4.1.6.9. VS_Ellipse

Отрисовка эллипса

```
bool VS_Ellipse( int nID, int nLeftRect, int nTopRect, int nRightRect, int nBottomRect, unsigned int unBoundColor, unsigned int unFillColor );
```

Параметры:

nID идентификатор окна.

nLeftRect X координата левого верхнего угла прямоугольника, описанного около эллипса.

nTopRect Y координата левого верхнего угла прямоугольника, описанного около эллипса.

nRightRect X координата правого нижнего угла прямоугольника, описанного около эллипса.

nBottomRect Y координата правого нижнего угла прямоугольника, описанного около эллипса.

unBoundColor цвет контура эллипса.

unFillColor цвет заполнителя эллипса.

Возвращаемое значение: в случае успеха функция возвращает true, в случае неудачи - false.

Описание: Функция отрисовывает эллипс, который вписан в прямоугольник, заданный координатами *nLeftRect*, *nTopRect*, *nRightRect*, *nBottomRect*. Цвет контура и цвет

заполнения задаётся так, как описано в функции [VS_SetPixel\(\)](#). Если в параметре `unFillColor` установлено значение [VS_NULL_COLOR](#), то будет нарисован только контур эллипса, без заполнения его внутренней части.

4.1.6.10. VS_EllipseEx

Отрисовка эллипса с возможностью установки стиля пера

```
bool VS_EllipseEx( int nID, int nLeftRect, int nTopRect, int nRightRect,
int nBottomRect, unsigned int unBoundColor, unsigned int unFillColor, int
nDashDot );
```

Параметры:

nID идентификатор окна.

nLeftRect X координата левого верхнего угла прямоугольника, описанного около эллипса.

nTopRect Y координата левого верхнего угла прямоугольника, описанного около эллипса.

nRightRect X координата правого нижнего угла прямоугольника, описанного около эллипса.

nBottomRect Y координата правого нижнего угла прямоугольника, описанного около эллипса.

unBoundColor цвет контура эллипса.

unFillColor цвет заполнителя эллипса.

nDashDot стиль пера, используемый при отрисовки контура. Для задания стиля используются [идентификаторы стиля пера](#), или 0 - если требуется отрисовка сплошной линией.

Возвращаемое значение: в случае успеха функция возвращает true, в случае неудачи - false.

Описание: Функция отрисовывает эллипс, который вписан в прямоугольник, заданный координатами `nLeftRect`, `nTopRect`, `nRightRect`, `nBottomRect`. Цвет контура и цвет заполнения задаётся так, как описано в функции [VS_SetPixel\(\)](#). Если в параметре `unFillColor` установлено значение [VS_NULL_COLOR](#), то будет нарисован только контур эллипса, без заполнения его внутренней части.

4.1.6.11. VS_EllipseF

Отрисовка эллипса

```
bool VS_EllipseF( int nID, float fltLeftRect, float fltTopRect, float
fltRightRect, float fltBottomRect, unsigned int unBoundColor, unsigned int
unFillColor );
```

Параметры:

nID идентификатор окна.

fltLeftRect X координата левого верхнего угла прямоугольника, описанного около эллипса.

fltTopRect Y координата левого верхнего угла прямоугольника, описанного около эллипса.

fltRightRect X координата правого нижнего угла прямоугольника, описанного около эллипса.

fltBottomRect Y координата правого нижнего угла прямоугольника, описанного около эллипса.

unBoundColor цвет контура эллипса.

unFillColor цвет заполнителя эллипса.

Возвращаемое значение: в случае успеха функция возвращает true, в случае неудачи - false.

Описание: Версия функции [VS_Ellipse\(\)](#) для чисел с плавающей точкой.

4.1.6.12. VS_EllipseFEx

Отрисовка эллипса с возможностью установки стиля пера

```
bool VS_EllipseFEx( int nID, float fltLeftRect, float fltTopRect, float
fltRightRect, float fltBottomRect, unsigned int unBoundColor, unsigned int
unFillColor, int nDashDot );
```

Параметры:

nID идентификатор окна.

fltLeftRect X координата левого верхнего угла прямоугольника, описанного около эллипса.

fltTopRect Y координата левого верхнего угла прямоугольника, описанного около эллипса.

fltRightRect X координата правого нижнего угла прямоугольника, описанного около эллипса.

fltBottomRect Y координата правого нижнего угла прямоугольника, описанного около эллипса.

unBoundColor цвет контура эллипса.

unFillColor цвет заполнителя эллипса.

nDashDot стиль пера, используемый при отрисовки контура. Для задания стиля используются [идентификаторы стиля пера](#), или 0 - если требуется отрисовка сплошной линией.

Возвращаемое значение: в случае успеха функция возвращает true, в случае неудачи - false.

Описание: Версия функции [VS_EllipseEx\(\)](#) для чисел с плавающей точкой.

4.1.6.13. VS_Line

Отрисовка линии

```
bool VS_Line( int nID, int nX1, int nY1, int nX2, int nY2, unsigned int
unColor );
```

Параметры:

nID идентификатор окна.
nX1 X координата начала линии.
nY1 Y координата начала линии.
nX2 X координата конца линии.
nY2 Y координата конца линии.
unColor цвет линии.

Возвращаемое значение: в случае успеха функция возвращает true, в случае неудачи - false.

Описание: Функция отрисовывает линию на изображении от точки с координатами [nX1, nY1] до точки с координатами [nX2, nY2] цветом, заданным в параметре unColor. Цвет линии задаётся так, как описано в функции [VS_SetPixel\(\)](#).

4.1.6.14. VS_LineEx

Отрисовка линии с возможностью установки стиля пера

```
bool VS_LineEx( int nID, int nX1, int nY1, int nX2, int nY2, unsigned
int unColor, int nDashDot );
```

Параметры:

nID идентификатор окна.
nX1 X координата начала линии.
nY1 Y координата начала линии.
nX2 X координата конца линии.
nY2 Y координата конца линии.
unColor цвет линии.
nDashDot стиль пера, используемый при отрисовки линий. Для задания стиля используются [идентификаторы стиля пера](#), или 0 - если требуется отрисовка сплошной линией.

Возвращаемое значение: в случае успеха функция возвращает true, в случае неудачи - false.

Описание: Функция отрисовывает линию на изображении от точки с координатами [nX1, nY1] до точки с координатами [nX2, nY2] цветом, заданным в параметре unColor. Цвет линии задаётся так, как описано в функции [VS_SetPixel\(\)](#).

4.1.6.15. VS_LineF

Отрисовка линии

```
bool VS_LineF( int nID, float fltX1, float fltY1, float fltX2, float fltY2,
unsigned int unColor );
```

Параметры:

nID идентификатор окна.
fltX1 X координата начала линии.
fltY1 Y координата начала линии.
fltX2 X координата конца линии.
fltY2 Y координата конца линии.
unColor цвет линии.

Возвращаемое значение: в случае успеха функция возвращает true, в случае неудачи - false.

Описание: Версия функции [VS_Line\(\)](#) для чисел с плавающей точкой.

4.1.6.16. VS_LineFEx

Отрисовка линии с возможностью установки стиля пера

```
bool VS_LineFEx( int nID, float fltX1, float fltY1, float fltX2, float fltY2,
unsigned int unColor, int nDashDot );
```

Параметры:

nID идентификатор окна.
fltX1 X координата начала линии.
fltY1 Y координата начала линии.
fltX2 X координата конца линии.
fltY2 Y координата конца линии.
unColor цвет линии.
nDashDot стиль пера, используемый при отрисовки линий. Для задания стиля используются [идентификаторы стиля пера](#), или 0 - если требуется отрисовка сплошной линией.

Возвращаемое значение: в случае успеха функция возвращает true, в случае неудачи - false.

Описание: Версия функции [VS_LineEx\(\)](#) для чисел с плавающей точкой.

4.1.6.17. VS_Pie

Отрисовка сектора

```
bool VS_Pie( int nID, int nLeftRect, int nTopRect, int nRightRect, int
nBottomRect, int nXRadial1, int nYRadial1, int nXRadial2, int nYRadial2,
unsigned int unBoundColor, unsigned int unFillColor );
```

Параметры:

nID идентификатор окна.

nLeftRect X координата левого верхнего угла прямоугольника, описанного около эллипса, на котором лежит сектор.

nTopRect Y координата левого верхнего угла прямоугольника, описанного около эллипса, на котором лежит сектор.

nRightRect X координата правого нижнего угла прямоугольника, описанного около эллипса, на котором лежит сектор.

nBottomRect Y координата правого нижнего угла прямоугольника, описанного около эллипса, на котором лежит сектор.

nXRadial1 X координата точки на начальном радиусе сектора.

nYRadial1 Y координата точки на начальном радиусе сектора.

nXRadial2 X координата точки на конечном радиусе сектора.

nYRadial2 Y координата точки на конечном радиусе сектора.

unBoundColor цвет контура сектора.

unFillColor цвет заполнителя сектора.

Возвращаемое значение: в случае успеха функция возвращает true, в случае неудачи - false.

Описание: Сектор является фигурой, образованной при пересечении эллипса с начальным и конечным радиусом. При вызове функции необходимо задать эллипс и указать ту его часть, которая должна содержаться в секторе. Для задания эллипса необходимо указать координаты прямоугольника, который описывается около эллипса с помощью параметров *nLeftRect*, *nTopRect*, *nRightRect* и *nBottomRect*. Функция рисует сектор от той точки, где начальный радиус пересекается с эллипсом, до точки его пересечения с конечным радиусом в направлении против часовой стрелки. Начальный радиус - это прямая, проходящая через центр эллипса и точку с координатами [*nXRadial1*, *nYRadial1*]. Конечный радиус - это прямая, проходящая через центр эллипса и точку с координатами [*nXRadial2*, *nYRadial2*]. Цвет контура и цвет заполнения задаётся так, как описано в функции [VS_SetPixel\(\)](#). Если в параметре *unFillColor* установлено значение [VS_NULL_COLOR](#), то будет нарисован только контур сектора, без заполнения его внутренней части.

4.1.6.18. VS_PieEx

Отрисовка сектора с возможностью установки стиля пера

```
bool VS_PieEx( int nID, int nLeftRect, int nTopRect, int nRightRect, int nBottomRect, int nXRadial1, int nYRadial1, int nXRadial2, int nYRadial2, unsigned int unBoundColor, unsigned int unFillColor, int nDashDot );
```

Параметры:

nID идентификатор окна.

nLeftRect X координата левого верхнего угла прямоугольника, описанного около эллипса, на котором лежит сектор.

nTopRect Y координата левого верхнего угла прямоугольника, описанного около эллипса, на котором лежит сектор.

nRightRect X координата правого нижнего угла прямоугольника, описанного около эллипса, на котором лежит сектор.

nBottomRect Y координата правого нижнего угла прямоугольника, описанного около эллипса, на котором лежит сектор.

nXRadial1 X координата точки на начальном радиусе сектора.

nYRadial1 Y координата точки на начальном радиусе сектора.

nXRadial2 X координата точки на конечном радиусе сектора.

nYRadial2 Y координата точки на конечном радиусе сектора.

unBoundColor цвет контура сектора.

unFillColor цвет заполнителя сектора.

nDashDot стиль пера, используемый при отрисовки контура. Для задания стиля используются [идентификаторы стиля пера](#), или 0 - если требуется отрисовка сплошной линией.

Возвращаемое значение: в случае успеха функция возвращает true, в случае неудачи - false.

Описание: Сектор является фигурой, образованной при пересечении эллипса с начальным и конечным радиусом. При вызове функции необходимо задать эллипс и указать ту его часть, которая должна содержаться в секторе. Для задания эллипса необходимо указать координаты прямоугольника, который описывается около эллипса с помощью параметров *nLeftRect*, *nTopRect*, *nRightRect* и *nBottomRect*. Функция рисует сектор от той точки, где начальный радиус пересекается с эллипсом, до точки его пересечения с конечным радиусом в направлении против часовой стрелки. Начальный радиус - это прямая, проходящая через центр эллипса и точку с координатами [*nXRadial1*, *nYRadial1*]. Конечный радиус - это прямая, проходящая через центр эллипса и точку с координатами [*nXRadial2*, *nYRadial2*]. Цвет контура и цвет заполнения задаётся так, как описано в функции [VS_SetPixel\(\)](#). Если в параметре *unFillColor* установлено значение [VS_NULL_COLOR](#), то будет нарисован только контур сектора, без заполнения его внутренней части.

4.1.6.19. VS_PieF

Отрисовка сектора

```
bool VS_PieF( int nID, float fltLeftRect, float fltTopRect, float fltRightRect,
float fltBottomRect, float fltXRadial1, float fltYRadial1, float fltXRadial2,
float fltYRadial2, unsigned int unBoundColor, unsigned int unFillColor );
```

Параметры:

nID идентификатор окна.

fltLeftRect X координата левого верхнего угла прямоугольника, описанного около эллипса, на котором лежит сектор.

fltTopRect Y координата левого верхнего угла прямоугольника, описанного около эллипса, на котором лежит сектор.

fltRightRect X координата правого нижнего угла прямоугольника, описанного около эллипса, на котором лежит сектор.

fltBottomRect Y координата правого нижнего угла прямоугольника, описанного около эллипса, на котором лежит сектор.

fltXRadial1 X координата точки на начальном радиусе сектора.

fltYRadial1 Y координата точки на начальном радиусе сектора.
fltXRadial2 X координата точки на конечном радиусе сектора.
fltYRadial2 Y координата точки на конечном радиусе сектора.
unBoundColor цвет контура сектора.
unFillColor цвет заполнителя сектора.

Возвращаемое значение: в случае успеха функция возвращает true, в случае неудачи - false.

Описание: Версия функции [VS_Pie\(\)](#) для чисел с плавающей точкой.

4.1.6.20. VS_PieFEx

Отрисовка сектора с возможностью установки стиля пера

```
bool VS_PieFEx( int nID, float fltLeftRect, float fltTopRect, float
fltRightRect, float fltBottomRect, float fltXRadial1, float fltYRadial1, float
fltXRadial2, float fltYRadial2, unsigned int unBoundColor, unsigned int
unFillColor, int nDashDot );
```

Параметры:

nID идентификатор окна.
fltLeftRect X координата левого верхнего угла прямоугольника, описанного около эллипса, на котором лежит сектор.
fltTopRect Y координата левого верхнего угла прямоугольника, описанного около эллипса, на котором лежит сектор.
fltRightRect X координата правого нижнего угла прямоугольника, описанного около эллипса, на котором лежит сектор.
fltBottomRect Y координата правого нижнего угла прямоугольника, описанного около эллипса, на котором лежит сектор.
fltXRadial1 X координата точки на начальном радиусе сектора.
fltYRadial1 Y координата точки на начальном радиусе сектора.
fltXRadial2 X координата точки на конечном радиусе сектора.
fltYRadial2 Y координата точки на конечном радиусе сектора.
unBoundColor цвет контура сектора.
unFillColor цвет заполнителя сектора.
nDashDot стиль пера, используемый при отрисовки контура. Для задания стиля используются [идентификаторы стиля пера](#), или 0 - если требуется отрисовка сплошной линией.

Возвращаемое значение: в случае успеха функция возвращает true, в случае неудачи - false.

Описание: Версия функции [VS_PieEx\(\)](#) для чисел с плавающей точкой.

4.1.6.21. VS_PolyBezier

Отрисовка кривой Безье

```
bool VS_PolyBezier( int nID, S_VS_Point *pPoints, int nPoints, unsigned
int unColor );
```

Параметры:

nID идентификатор окна.

pPoints массив точек.

nPoints количество точек.

unColor цвет кривой Безье.

Возвращаемое значение: в случае успеха функция возвращает true, в случае неудачи - false.

Описание: Функция отрисовывает кривую Безье по точкам, заданным в параметре pPoints. Для рисования кривой Безье в массиве точек должно содержаться, по крайней мере, координаты четырёх точек: первая является начальной точкой кривой, вторая и третья - контрольными точками, определяющими её форму, а четвёртая - конечной точкой. Чтобы нарисовать две соединяющиеся между собой кривой Безье нужно добавить к массиву точек только три точки. Начальная точка второй, дополнительной кривой, будет совпадать с конечной точкой первой кривой. Первая и вторая дополнительные точки будут контрольными точками второй кривой, а третья станет её конечной точкой. Цвет кривой задаётся так, как описано в функции [VS_SetPixel\(\)](#).

4.1.6.22. VS_PolyBezierEx

Отрисовка кривой Безье с возможностью установки стиля пера

```
bool VS_PolyBezierEx( int nID, S_VS_Point *pPoints, int nPoints,
unsigned int unColor, int nDashDot );
```

Параметры:

nID идентификатор окна.

pPoints массив точек.

nPoints количество точек.

unColor цвет кривой Безье.

nDashDot стиль пера, используемый при отрисовки линий. Для задания стиля используются [идентификаторы стиля пера](#), или 0 - если требуется отрисовка сплошной линией.

Возвращаемое значение: в случае успеха функция возвращает true, в случае неудачи - false.

Описание: Функция отрисовывает кривую Безье по точкам, заданным в параметре pPoints. Для рисования кривой Безье в массиве точек должно содержаться, по крайней мере, координаты четырёх точек: первая является начальной точкой кривой, вторая и третья - контрольными точками, определяющими её форму, а четвёртая - конечной точкой. Чтобы нарисовать две соединяющиеся между собой кривой

Безье нужно добавить к массиву точек только три точки. Начальная точка второй, дополнительной кривой, будет совпадать с конечной точкой первой кривой. Первая и вторая дополнительные точки будут контрольными точками второй кривой, а третья станет её конечной точкой. Цвет кривой задаётся так, как описано в функции [VS_SetPixel\(\)](#).

4.1.6.23. VS_PolyBezierF

Отрисовка кривой Безье

```
bool VS_PolyBezierF( int nID, S_VS_PointF *pPoints, int nPoints,
unsigned int unColor );
```

Параметры:

nID идентификатор окна.
pPoints массив точек.
nPoints количество точек.
unColor цвет кривой Безье.

Возвращаемое значение: в случае успеха функция возвращает true, в случае неудачи - false.

Описание: Версия функции [VS_PolyBezier\(\)](#) для чисел с плавающей точкой.

4.1.6.24. VS_PolyBezierFEx

Отрисовка кривой Безье с возможностью установки стиля пера

```
bool VS_PolyBezierFEx( int nID, S_VS_PointF *pPoints, int nPoints,
unsigned int unColor, int nDashDot );
```

Параметры:

nID идентификатор окна.
pPoints массив точек.
nPoints количество точек.
unColor цвет кривой Безье.
nDashDot стиль пера, используемый при отрисовки линий. Для задания стиля используются [идентификаторы стиля пера](#), или 0 - если требуется отрисовка сплошной линией.

Возвращаемое значение: в случае успеха функция возвращает true, в случае неудачи - false.

Описание: Версия функции [VS_PolyBezierEx\(\)](#) для чисел с плавающей точкой.

4.1.6.25. VS_Polygon

Отрисовка полигона

```
bool VS_Polygon( int nID, S_VS_Point *pPoints, int nPoints, unsigned int
unBoundColor, unsigned int unFillColor );
```

Параметры:

nID идентификатор окна.
pPoints точки (вершины) полигона.
nPoints количество вершин полигона.
unBoundColor цвет контура полигона.
unFillColor цвет заполнителя полигона.

Возвращаемое значение: в случае успеха функция возвращает true, в случае неудачи - false.

Описание: Полигон образуется из точек, заданных в параметре *pPoints*, которые являются вершинами полигона. Все вершины полигона последовательно соединяются линиями, причём последняя вершина соединяется с первой, образуя, таким образом, замкнутую фигуру. Внутреннее пространство полигона закрашивается цветом заполнителя. Цвет контура и цвет заполнения задаётся так, как описано в функции [VS_SetPixel\(\)](#). Если в параметре *unFillColor* установлено значение [VS_NULL_COLOR](#), то будет нарисован только контур полигона, без заполнения его внутренней части.

4.1.6.26. VS_PolygonEx

Отрисовка полигона с возможностью установки стиля пера

```
bool VS_PolygonEx( int nID, S_VS_Point *pPoints, int nPoints, unsigned
int unBoundColor, unsigned int unFillColor, int nDashDot );
```

Параметры:

nID идентификатор окна.
pPoints точки (вершины) полигона.
nPoints количество вершин полигона.
unBoundColor цвет контура полигона.
unFillColor цвет заполнителя полигона.
nDashDot стиль пера, используемый при отрисовки контура. Для задания стиля используются [идентификаторы стиля пера](#), или 0 - если требуется отрисовка сплошной линией.

Возвращаемое значение: в случае успеха функция возвращает true, в случае неудачи - false.

Описание: Полигон образуется из точек, заданных в параметре *pPoints*, которые являются вершинами полигона. Все вершины полигона последовательно соединяются линиями, причём последняя вершина соединяется с первой, образуя, таким образом, замкнутую фигуру. Внутреннее пространство полигона закрашивается цветом заполнителя. Цвет контура и цвет заполнения задаётся так, как описано в функции [VS_SetPixel\(\)](#). Если в параметре *unFillColor* установлено значение [VS_NULL_COLOR](#), то будет нарисован только контур полигона, без заполнения его внутренней части.

4.1.6.27. VS_PolygonF

Отрисовка полигона

```
bool VS_PolygonF( int nID, S_VS_PointF *pPoints, int nPoints, unsigned
int unBoundColor, unsigned int unFillColor );
```

Параметры:

nID идентификатор окна.
pPoints точки (вершины) полигона.
nPoints количество вершин полигона.
unBoundColor цвет контура полигона.
unFillColor цвет заполнителя полигона.

Возвращаемое значение: в случае успеха функция возвращает true, в случае неудачи - false.

Описание: Версия функции [VS_Polygon\(\)](#) для чисел с плавающей точкой.

4.1.6.28. VS_PolygonFEx

Отрисовка полигона с возможностью установки стиля пера

```
bool VS_PolygonFEx( int nID, S_VS_PointF *pPoints, int nPoints,
unsigned int unBoundColor, unsigned int unFillColor, int nDashDot );
```

Параметры:

nID идентификатор окна.
pPoints точки (вершины) полигона.
nPoints количество вершин полигона.
unBoundColor цвет контура полигона.
unFillColor цвет заполнителя полигона.
nDashDot стиль пера, используемый при отрисовки контура. Для задания стиля используются [идентификаторы стиля пера](#), или 0 - если требуется отрисовка сплошной линией.

Возвращаемое значение: в случае успеха функция возвращает true, в случае неудачи - false.

Описание: Версия функции [VS_PolygonEx\(\)](#) для чисел с плавающей точкой.

4.1.6.29. VS_Polyline

Отрисовка ломаной кривой

```
bool VS_Polyline( int nID, S_VS_Point *pPoints, int nPoints, unsigned int
unColor );
```

Параметры:

nID идентификатор окна.
pPoints массив точек.
nPoints количество точек.
unColor цвет ломаной кривой.

Возвращаемое значение: в случае успеха функция возвращает true, в случае неудачи - false.

Описание: Функция отрисовывает ломаную кривую по точкам, заданным в параметре *pPoints*. Все точки заданного массива точек последовательно соединяются между собой линиями цветом, заданным параметром *unColor*. Цвет задаётся так, как описано в функции [VS_SetPixel\(\)](#).

4.1.6.30. VS_PolylineEx

Отрисовка ломаной кривой с возможностью установки стиля пера

```
bool VS_PolylineEx( int nID, S_VS_Point *pPoints, int nPoints, unsigned
int unColor, int nDashDot );
```

Параметры:

nID идентификатор окна.
pPoints массив точек.
nPoints количество точек.
unColor цвет ломаной кривой.
nDashDot стиль пера, используемый при отрисовки линий. Для задания стиля используются [идентификаторы стиля пера](#), или 0 - если требуется отрисовка сплошной линией.

Возвращаемое значение: в случае успеха функция возвращает true, в случае неудачи - false.

Описание: Функция отрисовывает ломаную кривую по точкам, заданным в параметре *pPoints*. Все точки заданного массива точек последовательно соединяются между собой линиями цветом, заданным параметром *unColor*. Цвет задаётся так, как описано в функции [VS_SetPixel\(\)](#).

4.1.6.31. VS_PolylineF

Отрисовка ломаной кривой

```
bool VS_PolylineF( int nID, S_VS_PointF *pPoints, int nPoints, unsigned
int unColor );
```

Параметры:

nID идентификатор окна.

pPoints массив точек.
nPoints количество точек.
unColor цвет ломаной кривой.

Возвращаемое значение: в случае успеха функция возвращает true, в случае неудачи - false.

Описание: Версия функции [VS_Polyline\(\)](#) для чисел с плавающей точкой.

4.1.6.32. VS_PolylineFEx

Отрисовка ломаной кривой с возможностью установки стиля пера

```
bool VS_PolylineFEx( int nID, S_VS_PointF *pPoints, int nPoints,
unsigned int unColor, int nDashDot );
```

Параметры:

nID идентификатор окна.
pPoints массив точек.
nPoints количество точек.
unColor цвет ломаной кривой.
nDashDot стиль пера, используемый при отрисовки линий. Для задания стиля используются [идентификаторы стиля пера](#), или 0 - если требуется отрисовка сплошной линией.

Возвращаемое значение: в случае успеха функция возвращает true, в случае неудачи - false.

Описание: Версия функции [VS_PolylineEx\(\)](#) для чисел с плавающей точкой.

4.1.6.33. VS_Rectangle

Отрисовка прямоугольника

```
bool VS_Rectangle( int nID, int nLeftRect, int nTopRect, int nRightRect,
int nBottomRect, unsigned int unBoundColor, unsigned int unFillColor );
```

Параметры:

nID идентификатор окна.
nLeftRect X координата левого верхнего угла прямоугольника.
nTopRect Y координата левого верхнего угла прямоугольника.
nRightRect X координата правого нижнего угла прямоугольника.
nBottomRect Y координата правого нижнего угла прямоугольника.
unBoundColor цвет контура прямоугольника.
unFillColor цвет заполнения прямоугольника.

Возвращаемое значение: в случае успеха функция возвращает true, в случае неудачи - false.

Описание: Функция отрисовывает прямоугольник, заданный координатами `nLeftRect`, `nTopRect`, `nRightRect`, `nBottomRect`. Цвет контура и цвет заполнения задаётся так, как описано в функции [VS_SetPixel\(\)](#). Если в параметре `unFillColor` установлено значение [VS_NULL_COLOR](#), то будет нарисован только контур прямоугольника, без заполнения его внутренней части.

4.1.6.34. VS_RectangleEx

Отрисовка прямоугольника с возможностью установки стиля пера

```
bool VS_RectangleEx( int nID, int nLeftRect, int nTopRect, int
nRightRect, int nBottomRect, unsigned int unBoundColor, unsigned int
unFillColor, int nDashDot );
```

Параметры:

nID идентификатор окна.

nLeftRect X координата левого верхнего угла прямоугольника.

nTopRect Y координата левого верхнего угла прямоугольника.

nRightRect X координата правого нижнего угла прямоугольника.

nBottomRect Y координата правого нижнего угла прямоугольника.

unBoundColor цвет контура прямоугольника.

unFillColor цвет заполнения прямоугольника.

nDashDot стиль пера, используемый при отрисовки контура. Для задания стиля используются [идентификаторы стиля пера](#), или 0 - если требуется отрисовка сплошной линией.

Возвращаемое значение: в случае успеха функция возвращает true, в случае неудачи - false.

Описание: Функция отрисовывает прямоугольник, заданный координатами `nLeftRect`, `nTopRect`, `nRightRect`, `nBottomRect`. Цвет контура и цвет заполнения задаётся так, как описано в функции [VS_SetPixel\(\)](#). Если в параметре `unFillColor` установлено значение [VS_NULL_COLOR](#), то будет нарисован только контур прямоугольника, без заполнения его внутренней части.

4.1.6.35. VS_RectangleF

Отрисовка прямоугольника

```
bool VS_RectangleF( int nID, float fltLeftRect, float fltTopRect, float
fltRightRect, float fltBottomRect, unsigned int unBoundColor, unsigned int
unFillColor );
```

Параметры:

nID идентификатор окна.

fltLeftRect X координата левого верхнего угла прямоугольника.

fltTopRect Y координата левого верхнего угла прямоугольника.

fltRightRect X координата правого нижнего угла прямоугольника.

fltBottomRect Y координата правого нижнего угла прямоугольника.
unBoundColor цвет контура прямоугольника.
unFillColor цвет заполнения прямоугольника.

Возвращаемое значение: в случае успеха функция возвращает true, в случае неудачи - false.

Описание: Версия функции [VS_Rectangle\(\)](#) для чисел с плавающей точкой.

4.1.6.36. VS_RectangleFEx

Отрисовка прямоугольника с возможностью установки стиля пера

```
bool VS_RectangleFEx( int nID, float fltLeftRect, float fltTopRect, float
fltRightRect, float fltBottomRect, unsigned int unBoundColor, unsigned int
unFillColor, int nDashDot );
```

Параметры:

nID идентификатор окна.
fltLeftRect X координата левого верхнего угла прямоугольника.
fltTopRect Y координата левого верхнего угла прямоугольника.
fltRightRect X координата правого нижнего угла прямоугольника.
fltBottomRect Y координата правого нижнего угла прямоугольника.
unBoundColor цвет контура прямоугольника.
unFillColor цвет заполнения прямоугольника.
nDashDot стиль пера, используемый при отрисовки контура. Для задания стиля используются [идентификаторы стиля пера](#), или 0 - если требуется отрисовка сплошной линией.

Возвращаемое значение: в случае успеха функция возвращает true, в случае неудачи - false.

Описание: Версия функции [VS_RectangleEx\(\)](#) для чисел с плавающей точкой.

4.1.6.37. VS_SetPixel

Установка цвета пикселя

```
bool VS_SetPixel( int nID, int nX, int nY, unsigned int unColor );
```

Параметры:

nID идентификатор окна.
nX X координата пикселя на изображении.
nY Y координата пикселя на изображении.
unColor устанавливаемый цвет пикселя.

Возвращаемое значение: в случае успеха функция возвращает true, в случае неудачи - false.

Описание: Функция устанавливает цвет для элемента изображения с координатами [nX, nY]. Цвет пикселя задаётся в 32-х битовом слове. Цвет в этом слове определяется следующим образом: в первом (младшем) байте содержится интенсивность красного цвета, во втором байте содержится интенсивность зелёного цвета, в третьем байте содержится интенсивность голубого цвета, четвёртый байт не используется. Для формирования этого слова можно воспользоваться макросом RGB(), определённым в wingdi.h.

4.1.6.38. VS_SetPixelF

Установка цвета пикселя

```
bool VS_SetPixelF( int nID, float fltX, float fltY, unsigned int unColor );
```

Параметры:

nID идентификатор окна.
fltX X координата пикселя на изображении.
fltY Y координата пикселя на изображении.
unColor устанавливаемый цвет пикселя.

Возвращаемое значение: в случае успеха функция возвращает true, в случае неудачи - false.

Описание: Версия функции [VS_SetPixel\(\)](#) для чисел с плавающей точкой.

4.1.6.39. VS_TextOut

Функция вывода текста на изображение

```
bool VS_TextOut( int nID, int nX, int nY, int nFontSize, unsigned int unColor, const char *szText, ... );
```

Параметры:

nID идентификатор окна.
nX X координата на изображении для вывода текста.
nY Y координата на изображении для вывода текста.
nSize высота текста в пикселях.
unColor цвет выводимого текста.
szText строка со спецификаторами форматирования.
... аргументы.

Возвращаемое значение: в случае успеха функция возвращает true, в случае неудачи - false.

Описание: Функция формирует строку с учётом спецификаторов форматирования и аргументов и накладывает её на изображение, начиная с точки [nX, nY]. Информацию о спецификаторах форматирования можно найти в описании функции printf() стандартной

библиотеки ввода-вывода C. Функция VS_TextOut формирует строку так же, как функция printf(). Цвет текста задаётся так, как описано в функции [VS_SetPixel\(\)](#).

4.2. Структуры VShell

4.2.1. S_VS_MouseStatus

Задаёт состояние мыши

```
typedef struct _S_VS_MouseStatus{ int nID; int nX; int nY; int nScaleX; int nScaleY; int nKey; int nCursor; } S_VS_MouseStatus;
```

Члены структуры:

nID Идентификатор активного окна. Имеется в виду идентификатор, заданный пользователем при создании окна.

nX X координата мыши.

nY Y координата мыши.

nScaleX X координата мыши с учетом изменения масштаба, то есть X координата пикселя в изображении.

nScaleY Y координата мыши Y с учетом изменения масштаба, то есть Y координата пикселя в изображении.

nKey Состояние кнопок мыши (см. [идентификаторы состояния мыши](#)).

nCursor Тип текущего курсора (см. [идентификаторы курсоров мыши](#)).

4.2.2. S_VS_Pal

Задаёт цвет в палитре изображения

```
typedef struct _S_VS_Pal{ unsigned char ucBlue; unsigned char ucGreen; unsigned char ucRed; unsigned char ucReserved; } S_VS_Pal;
```

Члены структуры:

ucBlue интенсивность голубого цвета

ucGreen интенсивность зелёного цвета

ucRed интенсивность красного цвета

ucReserved не используется

4.2.3. S_VS_Point

Задаёт позицию точки на изображении в целых числах

```
typedef struct _S_VS_Point{ int nX; int nY; } S_VS_Point;
```

Члены структуры:

nX X координата точки

nY Y координата точки

4.2.4. S_VS_PointF

Задаёт позицию точки на изображении в числах с плавающей точках

```
typedef struct _S_VS_PointF{ float fltX; float fltY; } S_VS_PointF;
```

Члены структуры:

fltX X координата точки

fltY Y координата точки

4.3. Идентификаторы VShell

4.3.1. Идентификатор групповой отрисовки

Идентификатор	Значение	Описание
VS_DRAW_ALL	0x80000000	Константа для передачи в функцию VS_Draw , для перерисовки всех окон.

4.3.2. Идентификатор окна - источника изображений

Идентификатор	Значение	Описание
VS_SOURCE	0x80000000	Определяет работу с источником изображений.

4.3.3. Идентификатор отсутствия цвета

Идентификатор	Значение	Описание
VS_NULL_COLOR	0x80000000	Идентификатор для отрисовки замкнутых фигур. Определяет, что при отрисовке замкнутых фигур их внутренние области будут прозрачными, то есть будет нарисован только контур. Используется в функциях векторной графики.

4.3.4. Идентификаторы курсоров мыши

Идентификатор	Значение	Описание
VS_CURSOR_ARROW	1	Курсор "стрелка".
VS_CURSOR_MAGNIFY	2	Курсор "лупа".

Идентификатор	Значение	Описание
VS_CURSOR_DRAG	3	Курсор "рука".

4.3.5. Идентификаторы отображения векторов движения

Идентификатор	Значение	Описание
VS_NULL_VEC	0x80000000	Идентификатор несуществующего вектора движения. Используется как признак того, что данный блок не имеет вектора движения. Следует отличать такие векторы от нулевых векторов. Нулевые векторы всё же помечаются точкой цветом, который получается инверсией цвета векторов, а отсутствующие векторы никак не помечаются.
VS_DEF_VEC_COLOR	0x80000000	Константа для передачи в функцию VS_setVectors или VS_SetVectorsF , которая показывает, что векторы будут отображаться цветом по умолчанию (цвет по умолчанию можно задать из приложения).

4.3.6. Идентификаторы цветов

Идентификатор	Значение	Описание
VS_RED	0xFF0000	Красный
VS_ORANGERED	0xFF4500	Красно-оранжевый
VS_ORANGE	0xFFA500	Оранжевый
VS_GOLD	0xFFD700	Золотой
VS_YELLOW	0xFFFF00	Жёлтый
VS_YELLOWGREEN	0x9ACD32	Жёлто-зелёный
VS_GREENYELLOW	0xADFF2F	Зелёно-жёлтый
VS_GREEN	0x008000	Зелёный
VS_LIME	0x00FF00	Яркий зелёный
VS_LIMEGREEN	0x32CD32	Серо-зелёный
VS_CYAN	0x00FFFF	Голубой
VS_DARKBLUE	0x00008B	Тёмно-синий
VS_BLUE	0x0000FF	Синий
VS_BLUEVIOLET	0x8A2BE2	Сине-фиолетовый
VS_PURPLE	0x800080	Пурпурный
VS_MAGENTA	0xFF00FF	Магента
VS_BLACK	0x000000	Чёрный

Идентификатор	Значение	Описание
VS_WHITE	0xFFFFFF	Белый
VS_GRAY	0x808080	Серый

4.3.7. Идентификаторы состояния мыши

Идентификатор	Значение	Описание
VS_MOUSE_LBUTTON	1	Нажата левая кнопка мыши.
VS_MOUSE_RBUTTON	2	Нажата правая кнопка мыши.
VS_MOUSE_SHIFT	4	Нажата клавиша <Shift>.
VS_MOUSE_CONTROL	8	Нажата клавиша <Ctrl>.
VS_MOUSE_MBUTTON	16	Нажата средняя кнопка мыши.

4.3.8. Идентификаторы состояния программы

Идентификатор	Значение	Описание
VS_CLOSE	0	Программа закрыта пользователем (нажата кнопка закрытия окна в правом верхнем углу окна).
VS_FORWARD	1	Нажата кнопка "Запуск вперёд" или "Шаг вперёд". Изображения в последовательности загружаются последовательно в прямом порядке.
VS_BACK	2	Нажата кнопка "Запуск назад" или "Шаг назад". Изображения в последовательности загружаются последовательно в обратном порядке.
VS_STEP	4	Нажата кнопка "Шаг вперёд" или "Шаг назад". Выполняется пошаговая загрузка изображений.
VS_PAUSE	8	Нажата кнопка "Пауза". В каждой итерации загружается одно и то же изображение.

4.3.9. Идентификаторы стиля пера для векторной графики

Идентификатор	Значение	Описание
VS_DASH	1	Все линии векторной графики, в том числе контуры фигур будут отрисовываться длинными пунктирными линиями.
VS_DOT	2	Все линии векторной графики, в том числе контуры фигур будут отрисовываться короткими пунктирными линиями.

4.3.10. Идентификаторы типов изображений

Идентификатор	Значение	Описание
VS_RGB1	1	Изображение 1 бит на пиксель. Цвет определяется палитрой элементов S_VS_Pal (2 элемента). Каждый элемент изображения является индексом в палитре. По умолчанию для данных изображений используется палитра из чёрного и белого цветов.
VS_RGB4	2	Изображение 4 бита на пиксель. Цвет определяется палитрой элементов S_VS_Pal (16 элементов). Каждый элемент изображения является индексом в палитре. По умолчанию для данных изображений используется полутоновая палитра (16 градаций серого цвета).
VS_RGB8	3	Изображение 8 бит на пиксель. Цвет определяется палитрой элементов S_VS_Pal (256 элементов). Каждый элемент изображения является индексом в палитре. По умолчанию для данных изображений используется полутоновая палитра (256 градаций серого цвета).
VS_RGB16	4	Палитра не используется. Цвет пикселя задаётся в 16-ти битах: BGR 5-5-5 (старший бит не используется).
VS_RGB24	5	Палитра не используется. Цвет пикселя задаётся в 24-х битах: BGR 8-8-8.
VS_RGB32	6	Палитра не используется. Цвет пикселя задаётся в 32-х битах: BGR 8-8-8 (четвёртый байт не используется).
VS_RGB8_8	7	Изображение 8 битов на пиксель. Каждый элемент изображения хранится в 8-ми разрядном знаковом слове. Цвет определяется палитрой элементов S_VS_Pal (256 элементов). Каждый элемент изображения является индексом в палитре. По умолчанию для данных изображений используется полутоновая палитра (256 градаций серого цвета).
VS_RGB8_16	8	Изображение 8 битов на пиксель. Каждый элемент изображения хранится в 16-ти разрядном знаковом слове. Цвет определяется палитрой элементов S_VS_Pal (256 элементов). Каждый элемент изображения является индексом в палитре. По умолчанию для

Идентификатор	Значение	Описание
		данных изображений используется полутоновая палитра (256 градаций серого цвета). При отображении значения элементов усекаются до диапазона от -128 до 127.
VS_RGB8_32	9	Изображение 8 битов на пиксель. Каждый элемент изображения хранится в 32-х разрядном знаковом слове. Цвет определяется палитрой элементов S_VS_Pal (256 элементов). Каждый элемент изображения является индексом в палитре. По умолчанию для данных изображений используется полутоновая палитра (256 градаций серого цвета). При отображении значения элементов усекаются до диапазона от -128 до 127.

4.3.11. Максимальные значения

Идентификатор	Значение	Описание
VS_MAX_SLIDER	32	Максимальное количество доступных пользователю слайдеров.
VS_MAX_RADIO_GROUP	16	Максимальное количество доступных пользователю групп радио кнопок.
VS_MAX_RADIO	16	Максимальное количество радио кнопок в группе.
VS_MAX_CHECK_BOX	32	Максимальное количество доступных пользователю переключателей.
VS_MAX_EDIT	32	Максимальное количество доступных пользователю окон ввода (окна редактирования).

4.3.12. Тип для задания цвета для custom палитры

Идентификатор	Значение	Описание
VS_STOP_COLOR	unsigned int	32-битное слово состоящее из четырех байтов: [0:7] blue, [8:15] green, [16:23] red, [24:31] index. Передаётся массивом в функцию VS_CreateCustomPalette создания custom палитры. Для формирования цвета можно использовать макрос VS_MAKE_STOP_COLOR(index,color).

5. Ограничения в демонстрационной версии

В демонстрационной версии отсутствуют следующие возможности:

Сохранение изображений и последовательностей изображений.

Работа с цветными палитрами, в 1, 4 и 8 битовых изображениях. Доступна только полутоновая палитра.

Также в демо-версии при выводе изображений в правом верхнем углу дочернего окна выводится надпись "Demo".



НАУЧНО-ТЕХНИЧЕСКИЙ ЦЕНТР

НТЦ Модуль
А/Я 166, Москва, 125190, Россия
Тел: +7 (499) 152-9698
Факс: +7 (499) 152-4661
E-Mail: rusales@module.ru
WWW: <http://www.module.ru>

©НТЦ Модуль, 2009

Все права защищены.

Никакая часть информации, приведенная в данном документе, не может быть адаптирована или воспроизведена, кроме как согласно письменному разрешению владельцев авторских прав. НТЦ Модуль оставляет за собой право производить изменения как в описании, так и в самом продукте без дополнительных уведомлений. НТЦ Модуль не несет ответственности за любой ущерб, причиненный использованием информации в данном описании, ошибками или недосказанностью в описании, а также путем неправильного использования продукта.

Напечатано в России. Дата публикации: 18/04/2014