МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«МОСКОВСКИЙ ФИЗИКО-ТЕХНИЧЕСКИЙ ИНСТИТУТ
(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)»

НАУКА В РЕГИОНЫ

Учебное пособие к курсам дополнительного общего образования для первого–четвёртого годов обучения

Документация к библиотеке TX Library

Версия: 00173a, Ревизия: 168

Справочное руководство

Часть 1

Фонд развития Физтех-лицея и Физтех-школ

Иннопрактика

МФТИ
Долгопрудный, 2021

Автор:
Дединский Илья Рудольфович
Старший преподаватель
кафедры информатики МФТИ

СОДЕРЖАНИЕ

1. ФАЙЛЫ

1.1. Файл Example01.cpp

Простейший пример использования **TXLib**.

1.2. Файл Example02.cpp

Улучшенный пример использования **TXLib**.

1.3. Файл Example03.cpp

Пример с функциями, после рефакторинга ***примера 2***.

1.4. Файл Example04.cpp

Пример с функциями с параметрами, после рефакторинга ***примера 3***.

1.5. Файл Example05.cpp

Пример с циклами.

1.6. Файл Example06.cpp

Расширенный пример с циклами.

1.7. Файл MinecraftStory.cpp

Мультфильм «Minecraft Story».

Зачётная работа за 3 четверть 7 класса

1.8. Файл Movie.cpp

Мультфильм *«Путь программиста»*.

Зачётная работа при поступлении в профильную физмат-группу 7 класса с углублённым изучением программирования.

1.9. Файл PhongDemo.cpp

Этюд «Освещение по Фонгу».

1.10. Файл Tennis.cpp

Пример использования функций txLoadImage(), txDeleteDC(), txBitBlt(), txTransparentBlt(), txAlphaBlend().

1.11. Файл TestSpeed.cpp

TXLib Benchmarking.

1.12. Файл Tree.cpp

Этюд «Дерево».

1.13. Файл TXGL.cpp

TXLib OpenGL Demo.

1.14. Файл AR.cpp

TXLib AR;) Demo.

1.15. Файл TXLib.h

1.15.1. Классы

- class txAutoLock

Класс для автоматической блокировки и разблокировки критической секции.

- struct txDialog

Базовый класс для диалоговых окон.

- struct txDialog::Layout

Класс для описания элемента диалогового окна (контрола)

1.15.2. Макросы

- `#define _TX_VER`

Текущая версия библиотеки.

- `#define _TX_MODULE`

Имя модуля **TXLib**. Входит в диагностические сообщения.

- `#define __TX_COMPILER__`

Имя и версия текущего компилятора

- `#define _TX_BUILDMODE`

Имя режима сборки

- `#define __TX_FILELINE__`

Макрос, раскрывающийся в имя файла и номер строки файла, где он встретился.

- `#define __TX_FUNCTION__`

Имя текущей функции

- `#define MAX(a, b)`

Возвращает максимальное из двух чисел

- `#define MIN(a, b)`

Возвращает минимальное из двух чисел

- `#define ROUND(x)`

Округляет число до целого

- `#define _TX_DESTROY_3D`

Ну просто очень удобный макрос.

- `#define ZERO(type)`

Обнулитель типов, не имеющих конструкторов

- `#define TX_ASSERT(cond)`

Замена стандартного макроса `assert()`, с выдачей сообщения через `txMessageBox()`, консоль и `OutputDebugString()`.

- `#define asserted`

Выводит диагностическое сообщение в случае нулевого или ложного результата.

- `#define verified asserted`

For compatibility with assert macro.

- `#define verify`

Выполняет команду (вычисляет выражение) и проверяет результат.

- `#define TX_ERROR(msg)`

Выводит развёрнутое диагностическое сообщение.

- `#define TX_DEBUG_ERROR(...)`

Выводит развёрнутое диагностическое сообщение в отладочном режиме.

- `#define txStackBackTrace()`

Распечатывает текущий стек вызовов функций в консоли.

- `#define txGDI(command, dc)`

Вызов функции Win32 GDI с автоматической блокировкой и разблокировкой.

- `#define please`

Ещё парочка макросов.

- `#define _`

Макрос, позволяющий передать переменное число параметров в какой-либо другой макрос.

- `#define TX_COMMA`

Синоним макроса `_` (символ подчеркивания)

1.15.3. Внутренняя диагностика

- `#define _TX_ALLOW_TRACE`

Включает/отключает внутреннюю трассировку исполнения кода библиотеки.

- `#define TX_TRACE`

Трассирует исполнение кода через `OutputDebugString()`.

- *Макросы для построения статической карты сообщений (`Message Map`)* `#define TX_BEGIN_MESSAGE_MAP()`

Заголовок карты сообщений (`Message Map`).

- `#define TX_HANDLE(id)`

Заголовок обработчика сообщения (`Message handler`) карты сообщений.

- `#define TX_COMMAND_MAP`

Начало карты команд (`Command map`) в карте сообщений.

- `#define TX_END_MESSAGE_MAP`

Завершитель карты сообщений.

1.15.4. Экспериментальные отладочные макросы

- `#define __TX_DEBUG_MACROS` *(«Группа отладочных макросов»)*

Отладочная печать переменной во время вычисления выражения или участка кода во время его выполнения.

1.15.5. Функции

- `int random (int range) tx_deprecated`

Генератор случайных чисел

- `double random (double left, double right) tx_deprecated`

Генератор случайных чисел

- `template<typename Tx, typename Ta, typename Tb > bool In (Tx x, Ta a, Tb b) tx_deprecated`

Проверка, находится ли параметр x внутри замкнутого интервала $[a; b]$.

- `void tx_fpreset ()`

Переинициализирует математический сопроцессор

- `double txSqr (double x)`

Очень удобное возведение числа в квадрат.

- `void txDump (const void *address, const char name[]="_txDump()", bool pause=true)`

Распечатывает дамп области памяти в консоли.

- `int txRegQuery (const char *keyName, const char *valueName, void *value, size_t szValue)`

Читает информацию из реестра Windows.

- `int txSetLocale (int codepage=_TX_CODEPAGE, const char locale[]=_TX_LOCALE, const wchar_t wLocale[]=_TX_WLOCALE)`

Смена кодовой страницы консоли и локали стандартной библиотеки **C++**.

1.15.6. Инициализация библиотеки

- `HWND txCreateWindow (double sizeX, double sizeY, bool centered=true)`

Создание окна рисования

- `HDC & txDC ()`

Возвращает холст (дескриптор контекста рисования, HDC) окна рисования **TXLib**.

- `RGBQUAD * txVideoMemory ()`

Возвращает буфер памяти, связанный с холстом (HDC) **TXLib**.

- `bool txSetDefaults (HDC dc=txDC())`

Установка параметров рисования по умолчанию.

- `bool txOK ()`

Проверка правильности работы библиотеки

- `POINT txGetExtent (HDC dc=txDC())`

Возвращает размер окна, картинки или холста в виде структуры `POINT`.

- `int txGetExtentX (HDC dc=txDC())`

Возвращает ширину окна или холста.

- `int txGetExtentY (HDC dc=txDC())`

Возвращает высоту окна или холста.

- `HWND txWindow ()`

Возвращает дескриптор окна рисования

- `const char * txVersion ()`

Возвращает строку с информацией о текущей версии библиотеки.

- `unsigned txVersionNumber ()`

Возвращает номер версии библиотеки.

- `const char * txGetModuleFileName (bool fileNameOnly=true)`

Возвращает имя исполняемого файла или изначальный заголовок окна **TXLib**.

1.15.7. Рисование фигур

- `bool txClear (HDC dc=txDC())`

Стирает холст текущим цветом заполнения.

- `bool txSetPixel (double x, double y, COLORREF color, HDC dc=txDC())`

Рисует пиксель (точку на экране).

- `COLORREF txGetPixel (double x, double y, HDC dc=txDC())`

Возвращает текущий цвет точки (пикселя) на экране.

- `bool txLine (double x0, double y0, double x1, double y1, HDC dc=txDC())`

Рисует линию.

- `bool txRectangle (double x0, double y0, double x1, double y1, HDC dc=txDC())`

Рисует прямоугольник.

- `bool txPolygon (const POINT points[], int numPoints, HDC dc=txDC())`

Рисует ломаную линию или многоугольник.

- `bool txEllipse (double x0, double y0, double x1, double y1, HDC dc=txDC())`

Рисует эллипс.

- `bool txCircle (double x, double y, double r)`

Рисует окружность или круг.

- `bool txArc (double x0, double y0, double x1, double y1, double startAngle, double totalAngle, HDC dc=txDC())`

Рисует дугу эллипса.

- `bool txPie (double x0, double y0, double x1, double y1, double startAngle, double totalAngle, HDC dc=txDC())`

Рисует сектор эллипса.

- `bool txChord (double x0, double y0, double x1, double y1, double startAngle, double totalAngle, HDC dc=txDC())`

Рисует хорду эллипса.

- `bool txFloodFill (double x, double y, COLORREF color=TX_TRANSPARENT, DWORD mode=FLOODFILLSURFACE, HDC dc=txDC())`

Заливает произвольный контур текущим цветом заполнения.

- `bool txTriangle (double x1, double y1, double x2, double y2, double x3, double y3)`

Функция, которая должна бы рисовать треугольник.

- `void txDrawMan (int x, int y, int sizeX, int sizeY, COLORREF color, double handL, double handR, double twist, double head, double eyes, double wink, double crazy, double smile, double hair, double wind)`

Рисует человечка.

1.15.8. Работа с текстом

- `bool txTextOut (double x, double y, const char text[], HDC dc=txDC())`

Рисует текст.

- `bool txDrawText (double x0, double y0, double x1, double y1, const char text[], unsigned format=DT_CENTER|DT_VCENTER|DT_WORDBREAK|DT_WORD_ELLIPSIS, HDC dc=txDC())`

Рисует текст, размещённый в прямоугольной области.

- `HFONT txSelectFont (const char name[], double sizeY, double sizeX=-1, int bold=FW_DONTCARE, bool italic=false, bool underline=false, bool strikeout=false, double angle=0, HDC dc=txDC())`

Выбирает текущий шрифт, его размер и другие атрибуты.

- `SIZE txGetTextExtent (const char text[], HDC dc=txDC())`

Вычисляет размеры текстовой надписи.

- `int txGetTextExtentX (const char text[], HDC dc=txDC())`

Вычисляет ширину текстовой надписи.

- `int txGetTextExtentY (const char text[], HDC dc=txDC())`

Вычисляет высоту текстовой надписи.

- `unsigned txSetTextAlign (unsigned align=TA_CENTER|TA_BASELINE, HDC dc=txDC())`

Устанавливает текущее выравнивание текста (влево/вправо/по центру).

- `LOGFONT * txFontExist (const char name[])`

Ищет шрифт по его названию.

1.15.9. Рисование в памяти (на «виртуальном холсте») и загрузка изображений

- `HDC txCreateCompatibleDC (double sizeX, double sizeY, HBITMAP bitmap=NULL, RGBQUAD **pixels=NULL)`

Создаёт дополнительный холст (контекст рисования, `Device Context, DC`) в памяти.

- `HDC txCreateDIBSection (double sizeX, double sizeY, RGBQUAD **pixels=NULL)`

Создаёт аппаратно-независимый дополнительный холст (контекст рисования, `Device Context, DC`) в памяти с возможностью прямого доступа к нему, как к массиву.

- `HDC txLoadImage (const char filename[], int sizeX=0, int sizeY=0, unsigned imageFlags=IMAGE_BITMAP, unsigned loadFlags=LR_LOADFROMFILE)`

Загружает из файла изображение в формате BMP. Делает это довольно медленно.

- `bool txDeleteDC (HDC dc)`

Уничтожает холст (контекст рисования, `DC`) в памяти.

- `bool txBitBlt (HDC destImage, double xDest, double yDest, double width, double height, HDC sourceImage, double xSource=0, double ySource=0, unsigned operation=SRCCOPY)`

Копирует изображение с одного холста (контекста рисования, `DC`) на другой.

- `bool txBitBlt (double xDest, double yDest, HDC sourceImage, double xSource=0, double ySource=0)`

Копирует изображение на экран.

- `bool txTransparentBlt (HDC destImage, double xDest, double yDest, double width, double height, HDC sourceImage, double xSource=0, double ySource=0, COLORREF transColor=TX_BLACK)`

Копирует изображение с одного холста (контекста рисования, `DC`) на другой с учётом прозрачности.

- `bool txTransparentBlt (double xDest, double yDest, HDC sourceImage, COLORREF transColor=TX_BLACK, double xSource=0, double ySource=0)`

Копирует изображение на экран с учётом прозрачности.

- `bool txAlphaBlend (HDC destImage, double xDest, double yDest, double width, double height, HDC sourceImage, double xSource=0, double ySource=0, double alpha=1.0)`

Копирует изображение с одного холста (контекста рисования, `DC`) на другой с учётом полупрозрачности.

- `bool txAlphaBlend (double xDest, double yDest, HDC sourceImage, double xSource=0, double ySource=0, double alpha=1.0)`

Копирует изображение на экран с учётом полупрозрачности.

- `HDC txUseAlpha (HDC image)`

Пересчитывает цвета пикселей с учётом прозрачности (переводит цвета в формат Premultiplied Alpha).

- `bool txSaveImage (const char filename[], HDC dc=txDC())`

Сохраняет в файл изображение в формате BMP.

1.15.10. Вспомогательные функции

- `double txSleep (double time=0)`

Задерживает выполнение программы на определённое время.

- `int txBegin ()`

Блокирует обновление изображения окна, во избежание мигания.

- `int txEnd ()`

Разблокирует обновление окна, заблокированное функцией `txBegin()`.

- `void txRedrawWindow ()`

Обновляет изображение в окне **TXLib** вручную.

- `int txUpdateWindow (int update=true)`

Разрешает или запрещает автоматическое обновление изображения в окне.

- `bool txSelectObject (HGDIOBJ obj, HDC dc=txDC())`

Устанавливает текущий активный объект GDI.

- `bool txIDontWantToHaveAPauseAfterMyProgramBeforeTheWindowWillClose_AndIWillNotBeAskingWhereIsMyPicture ()`

Делает нечто иногда удобное. *См. название функции.*

- `bool txDestroyWindow (HWND wnd=txWindow())`

Уничтожает окно.

- `double txQueryPerformance ()`

Оценивает скорость работы компьютера.

- `double txGetFPS (int minFrames=txFramesToAverage)`

Выдаёт количество кадров (вызовов этой функции) в секунду.

1.15.11. Работа с мышью

- `POINT txMousePos ()`

Возвращает позицию мыши.

- `double txMouseX ()`

Возвращает X-Координату мыши.

- `double txMouseY ()`

Возвращает Y-Координату мыши.

- `unsigned txMouseButtons ()`

Возвращает состояние кнопок мыши.

- `Mouse & txCatchMouse (bool shouldEat=true)`

Ловит Мышь!

1.15.12. Функции консоли

- `unsigned txSetConsoleAttr (unsigned colors=0x07)`

Устанавливает цветовые атрибуты консоли.

- `unsigned txGetConsoleAttr ()`

Возвращает текущие цветовые атрибуты консоли.

- `bool txClearConsole ()`

Стирает текст консоли.

- `POINT txSetConsoleCursorPos (double x, double y)`

Устанавливает позицию мигающего курсора консоли.

- `POINT txGetConsoleCursorPos ()`

Возвращает позицию мигающего курсора консоли.

- `POINT txGetConsoleExtent ()`

Возвращает размер консоли.

- `POINT txGetConsoleFontSize ()`

Возвращает размеры шрифта консоли.

- `bool txTextCursor (bool blink=true)`

Запрещает или разрешает рисование мигающего курсора в окне.

- `bool In (const POINT &pt, const RECT &rect) tx_deprecated`

Проверка, находится ли точка pt внутри прямоугольника rect.

1.15.13. Очень служебные функции

- `WNDPROC txSetWindowsHook (WNDPROC wndProc=NULL)`

Устанавливает альтернативную функцию обработки оконных сообщений Windows (оконную функцию) для окна **TXLib**.

- `bool txLock (bool wait=true)`

Блокировка холста (контекста рисования).

- `bool txUnlock ()`

Разблокировка холста

- *Пример использования класса диалога: функция* `txInputBox()const char * txInputBox`

(const char *text=NULL, const char *caption=NULL, const char *input=NULL)

Ввод строки в отдельном окне.

1.15.14. Переменные

- const double txPI

Число Пи

1.15.15. Установка цветов и режимов рисования

- enum txColors { TX_BLACK = RGB (0, 0, 0), TX_BLUE = RGB (0, 0, 128), TX_GREEN = RGB (0, 128, 0), TX_CYAN = RGB (0, 128, 128), TX_RED = RGB (128, 0, 0), TX_MAGENTA = RGB (128, 0, 128), TX_BROWN = RGB (128, 128, 0), TX_ORANGE = RGB (255, 128, 0), TX_GRAY = RGB (160, 160, 160), TX_DARKGRAY = RGB (128, 128, 128), TX_LIGHTGRAY = RGB (192, 192, 192), TX_LIGHTBLUE = RGB (0, 0, 255), TX_LIGHTGREEN = RGB (0, 255, 128), TX_LIGHTCYAN = RGB (0, 255, 255), TX_LIGHTRED = RGB (255, 0, 128), TX_LIGHTMAGENTA = RGB (255, 0, 255), TX_PINK = RGB (255, 128, 255), TX_YELLOW = RGB (255, 255, 128), TX_WHITE = RGB (255, 255, 255), TX_TRANSPARENT = 0xFFFFFFFF, TX_NULL = TX_TRANSPARENT, TX_HUE = 0x04000000, TX_SATURATION = 0x05000000, TX_LIGHTNESS = 0x06000000 }

Названия предопределённых цветов.

- COLORREF RGB (int red, int green, int blue)

Создаёт (смешивает) цвет из трёх базовых цветов (компонент).

- HPEN txSetColor (COLORREF color, double thickness=1, HDC dc=txDC())

Устанавливает текущий цвет и толщину линий, цвет текста.

- COLORREF txGetColor (HDC dc=txDC())

Возвращает текущий цвет линий и текста.

- `HBRUSH txSetFillColor (COLORREF color, HDC dc=txDC())`

Устанавливает текущий цвет заполнения фигур.

- `COLORREF txGetFillColor (HDC dc=txDC())`

Возвращает текущий цвет заполнения фигур.

- `unsigned txExtractColor (COLORREF color, COLORREF component)`

Извлекает цветовую компоненту (цветовой канал) из смешанного цвета.

- `COLORREF txRGB2HSL (COLORREF rgbColor)`

Преобразует цвет из формата RGB в формат HSL.

- `COLORREF txHSL2RGB (COLORREF hslColor)`

Преобразует цвет из формата HSL в формат RGB.

1.15.16. Другие полезные функции, не связанные с рисованием

- `#define sizearr(arr) ( sizeof (arr) / sizeof ((arr)[0]) )`

Вычисление размера массива в элементах

- `bool txPlaySound (const char filename[]=NULL, DWORD mode=SND_ASYNC)`

Воспроизводит звуковой файл.

- `int txSpeak (const char *text, ...)`

Читает ~~мысли~~ текст вслух.

- `intptr_t txPlayVideo (int x, int y, int width, int height, const char fileName[], double zoom=0, double gain=1, HWND wnd=txWindow())`

Проигрывает видео.

- `intptr_t txPlayVideo (const char fileName[], double zoom=0, double gain=1, HWND wnd=txWindow())`

Проигрывает видео.

- `int txMessageBox (const char text[]="Myaxxaxa! :)", const char header[]="TXLib сообщает", unsigned flags=MB_ICONINFORMATION|MB_OKCANCEL)`

Выводит сообщение в окне с помощью `MessageBox`.

- `bool txGetAsyncKeyState (int key)`

Проверяет, нажата ли указанная клавиша.

- `bool txNotifyIcon (unsigned flags, const char title[], const char format[], ...)`

Выводит всплывающее сообщение в системном трее.

- `int txOutputDebugPrintf (const char format[], ...)`

Выводит сообщение в отладчике.

- `template<typename T, typename... ArgsT> int txPrintf (const char *format, ArgsT ...args)`

Добрый дядюшка Принтф. Теперь шаболонный.

- `template<typename T, typename... ArgsT> int txPrintf (std::ostringstream &stream, const char *format, ArgsT...args)`

Печатает в строковый поток вывода, как `sprintf()`.

- `template<typename T, typename... ArgsT> int txPrintf (char buffer[], size_t size, const char *format, ArgsT...args)`

Печатает в строковый буфер, как `sprintf()`.

- `template<typename... ArgsT> std::string txFormat (const char *format, ArgsT...args)`

Форматирует строку, как `sprintf()`.

- `#define tx_auto_func(func)`

Автоматический вызов функции при завершении другой функции (аналог `__finally`)

- `std::string txDemangle (const char *mangledName)`

Преобразует декорированное имя **C++** в название типа.

1.15.17. Настроечные константы и переменные

- `#define _TX_NOINIT`

Запрет ранней инициализации **TXLib.**

- `#define TX_USE_SFML`

Макрос, разрешающий использовать **TXLib** вместе с графической библиотекой SFML

- `#define _TX_EXCEPTIONS_LIMIT 16`

Максимальное количество исключений в программе.

- `#define _TX_FATAL_EXCEPTIONS_LIMIT 16`

Максимальное количество фатальных исключений.

- `#define _TX_FULL_STACKTRACE`

Если определено, не исключать адреса без отладочной информации из трассировок стека.

- `#define _TX_WAITABLE_PARENTS`

Список запускающих программ, которые ждут нажатия клавиши после завершения процесса **TXLib**.

- `#define _TX_ALLOW_KILL_PARENT true`

Разрешать принудительное завершение вызывающих программ, ждущих нажатия клавиш после завершения **TXLib**.

- `#define TX_COMPILED`

Макросы для поддержки прекомпиляции **TX Library**.

- `char _txLogName [MAX_PATH] = ""`

Имя лог-файла **TXLib**.

- `int _txConsoleMode = SW_HIDE`

Режим отображения консольного окна. Допустимы любые флаги функции `ShowWindow`.

- `int _txWindowStyle = WS_POPUP | WS_BORDER | WS_CAPTION | WS_SYSMENU`

Стиль графического окна библиотеки.

- `const char * _txConsoleFont = "Lucida Console"`

Шрифт консоли

- `unsigned _txCursorBlinkInterval = 500`

Интервал мигания курсора консоли (мс)

- `unsigned _txWindowUpdateInterval = 25`

Интервал обновления холста (мс)

- `const int _TX_TIMEOUT = 1000`

Таймаут операций ожидания событий (мс)

- `bool(* _txSwapBuffers )(HDC dest, int xDest, int yDest, int wDest, int hDest, HDC src, int xSrc, int ySrc, int wSrc, int hSrc, DWORD rOp) = NULL`

Указатель на функцию, выводящую изображение непосредственно в окно **TXLib** во время обработки `WM_PAINT`.

- `const unsigned _TX_BUFSIZE = 1024`

Размеры внутренних статических строковых буферов **TXLib**.

- `const unsigned _TX_BIGBUFSIZE = _TX_BUFSIZE * 2`

Размеры больших статических буферов.

- `const unsigned _TX_HUGEBUFSIZE = _TX_BUFSIZE * 20`

Размеры очень больших статических буферов.

- `const unsigned _TX_STACKSIZE = 64 * 1024`

Минимальный размер стека для потоков программы.

- `int _txWatchdogTimeout = 10*_TX_TIMEOUT`

Лимит времени на завершение программы, начиная от завершения функции `main()` или от вызова `exit()`, в мс.

1.16. Файл TXWave.h

1.16.1. Классы

- `union txWaveSample_t`

Тип данных, использующийся для внутреннего представления звуков согласно формату `txWaveFormat`.

1.16.2. Макросы

- `#define CALLOC(type, size)`

Выделяет блок динамической памяти через `calloc` с автоматическим преобразованием типа указателя.

- `#define FREE(ptr)`

Освобождает динамическую память и обнуляет указатель на неё.

1.16.3. Определения типов

- `typedef std::vector< txWaveSample_t > txWaveData_t`

Тип, использующийся для буферов данных.

- `typedef bool MonitorProc_t (HWAVEIN waveIn, txWaveData_t &data, void *userData)`

Тип функции-монитора для функции `txWaveIn()`.

1.16.4. Функции

- `HWAVEOUT txWaveOut (int timeMs=-INT_MAX, double freqL=0, double volL=50, double freqR=-1, double volR=-1, int loops=1, const txWaveData_t &data=txWaveData_t())`

Проигрывает звук через звуковую карту.

- `HWAVEOUT txWaveOut (const txWaveData_t &data, int loops=1)`

Проигрывает подготовленный или загруженный буфер через звуковую карту.

- `bool MonitorProc (HWAVEIN waveIn, txWaveData_t &data, void *userData)`

Функция-монитор, регулярно вызывающаяся при записи звука.

- `txWaveData_t txWaveIn (int timeMs, MonitorProc_t *monitorProc=NULL, void *monitorData=NULL, unsigned frameTime=0)`

Записывает звук со звуковой карты.

- `unsigned long txWaveGetPosition (void *wave)`

Возвращает текущую позицию воспроизведения или записи.

- `txWaveData_t txWaveLoadWav (const char filename[])`

Загружает звуковые данные из WAV-файла.

- `bool txWaveSaveWav (const txWaveData_t &data, const char filename[])`

Сохраняет звуковые данные в WAV-файле.

- `void * operator new (size_t size, int)`

Выделяет блок динамической памяти через `new` с обнулением его содержимого перед вызовом конструктора.

- `void * operator new (size_t size, size_t items, int)`

Выделяет блок динамической памяти через `new[]` с обнулением содержимого перед вызовом конструкторов.

1.16.5. Переменные

- `const double txWaveSampleRate = 44.100`

Скорость аудиопотока для `TXWave` в семплах на 1 миллисекунду.

- `const WAVEFORMATEX txWaveFormat`

Формат аудиоданных для `TXWave`.

- `const double txWaveVolMax`

Максимальная громкость в `txWaveSample_t`, согласно формату `txWaveFormat`.

2. РИСОВАНИЕ

2.1. Инициализация библиотеки

- `HWND txCreateWindow (double sizeX, double sizeY, bool centered=true)`

Создание окна рисования

- `HDC & txDC ()`

Возвращает холст (дескриптор контекста рисования, HDC) окна рисования **TXLib**.

- `RGBQUAD * txVideoMemory ()`

Возвращает буфер памяти, связанный с холстом (HDC) **TXLib**.

- `bool txSetDefaults (HDC dc=txDC())`

Установка параметров рисования по умолчанию.

- `bool txOK ()`

Проверка правильности работы библиотеки

- `POINT txGetExtent (HDC dc=txDC())`

Возвращает размер окна, картинки или холста в виде структуры POINT.

- `int txGetExtentX (HDC dc=txDC())`

Возвращает ширину окна или холста.

- `int txGetExtentY (HDC dc=txDC())`

Возвращает высоту окна или холста.

- `HWND txWindow ()`

Возвращает дескриптор окна рисования

- `const char * txVersion ()`

Возвращает строку с информацией о текущей версии библиотеки.

- `unsigned txVersionNumber ()`

Возвращает номер версии библиотеки.

- `const char * txGetModuleFileName (bool fileNameOnly=true)`

Возвращает имя исполняемого файла или изначальный заголовок окна **TXLib**.

2.2. Установка цветов и режимов рисования

- `enum txColors { TX_BLACK = RGB (0, 0, 0), TX_BLUE = RGB (0, 0, 128), TX_GREEN = RGB (0, 128, 0), TX_CYAN = RGB (0, 128, 128), TX_RED = RGB (128, 0, 0), TX_MAGENTA = RGB (128, 0, 128), TX_BROWN = RGB (128, 128, 0), TX_ORANGE = RGB (255, 128, 0), TX_GRAY = RGB (160, 160, 160), TX_DARKGRAY = RGB (128, 128, 128), TX_LIGHTGRAY = RGB (192, 192, 192), TX_LIGHTBLUE = RGB (0, 0, 255), TX_LIGHTGREEN = RGB (0, 255, 128), TX_LIGHTCYAN = RGB (0, 255, 255), TX_LIGHTRED = RGB (255, 0, 128), TX_LIGHTMAGENTA = RGB (255, 0, 255), TX_PINK = RGB (255, 128, 255), TX_YELLOW = RGB (255, 255, 128), TX_WHITE = RGB (255, 255, 255), TX_TRANSPARENT = 0xFFFFFFFF, TX_NULL = TX_TRANSPARENT, TX_HUE = 0x04000000, TX_SATURATION = 0x05000000, TX_LIGHTNESS = 0x06000000 }`

Названия предопределённых цветов.

- `COLORREF RGB (int red, int green, int blue)`

Создаёт (смешивает) цвет из трёх базовых цветов (компонент).

- `HPEN txSetColor (COLORREF color, double thickness=1, HDC dc=txDC())`

Устанавливает текущий цвет и толщину линий, цвет текста.

- `COLORREF txGetColor (HDC dc=txDC())`

Возвращает текущий цвет линий и текста.

- `HBRUSH txSetFillColor (COLORREF color, HDC dc=txDC())`

Устанавливает текущий цвет заполнения фигур.

- `COLORREF txGetFillColor (HDC dc=txDC())`

Возвращает текущий цвет заполнения фигур.

- `unsigned txExtractColor (COLORREF color, COLORREF component)`

Извлекает цветовую компоненту (цветовой канал) из смешанного цвета.

- `COLORREF txRGB2HSL (COLORREF rgbColor)`

Преобразует цвет из формата RGB в формат HSL.

- `COLORREF txHSL2RGB (COLORREF hslColor)`

Преобразует цвет из формата HSL в формат RGB.

2.3. Рисование фигур

- `bool txClear (HDC dc=txDC())`

Стирает холст текущим цветом заполнения.

- `bool txSetPixel (double x, double y, COLORREF color, HDC dc=txDC())`

Рисует пиксель (точку на экране).

- `COLORREF txGetPixel (double x, double y, HDC dc=txDC())`

Возвращает текущий цвет точки (пикселя) на экране.

- `bool txLine (double x0, double y0, double x1, double y1, HDC dc=txDC())`

Рисует линию.

- `bool txRectangle (double x0, double y0, double x1, double y1, HDC dc=txDC())`

Рисует прямоугольник.

- `bool txPolygon (const POINT points[], int numPoints, HDC dc=txDC())`

Рисует ломаную линию или многоугольник.

- `bool txEllipse (double x0, double y0, double x1, double y1, HDC dc=txDC())`

Рисует эллипс.

- `bool txCircle (double x, double y, double r)`

Рисует окружность или круг.

- `bool txArc (double x0, double y0, double x1, double y1, double startAngle, double totalAngle, HDC dc=txDC())`

Рисует дугу эллипса.

- `bool txPie (double x0, double y0, double x1, double y1, double startAngle, double totalAngle, HDC dc=txDC())`

Рисует сектор эллипса.

- `bool txChord (double x0, double y0, double x1, double y1, double startAngle, double totalAngle, HDC dc=txDC())`

Рисует хорду эллипса.

- `bool txFloodFill (double x, double y, COLORREF color=TX_TRANSPARENT, DWORD mode=FLOODFILLSURFACE, HDC dc=txDC())`

Заливает произвольный контур текущим цветом заполнения.

- `bool txTriangle (double x1, double y1, double x2, double y2, double x3, double y3)`

Функция, которая должна бы рисовать треугольник.

- `void txDrawMan (int x, int y, int sizeX, int sizeY, COLORREF color, double handL, double handR, double twist, double head, double eyes, double wink, double crazy, double smile, double hair, double wind)`

Рисует человечка.

2.4. Работа с текстом

- `bool txTextOut (double x, double y, const char text[], HDC dc=txDC())`

Рисует текст.

- `bool txDrawText (double x0, double y0, double x1, double y1, const char text[], unsigned format=DT_CENTER|DT_VCENTER|DT_WORDBREAK|DT_WORD_ELLIPSIS, HDC dc=txDC())`

Рисует текст, размещённый в прямоугольной области.

- `HFONT txSelectFont (const char name[], double sizeY, double sizeX=-1, int bold=FW_DONTCARE, bool italic=false, bool underline=false, bool strikeout=false, double angle=0, HDC dc=txDC())`

Выбирает текущий шрифт, его размер и другие атрибуты.

- `SIZE txGetTextExtent (const char text[], HDC dc=txDC())`

Вычисляет размеры текстовой надписи.

- `int txGetTextExtentX (const char text[], HDC dc=txDC())`

Вычисляет ширину текстовой надписи.

- `int txGetTextExtentY (const char text[], HDC dc=txDC())`

Вычисляет высоту текстовой надписи.

- `unsigned txSetTextAlign (unsigned align=TA_CENTER|TA_BASELINE, HDC dc=txDC())`

Устанавливает текущее выравнивание текста (влево/вправо/по центру).

- `LOGFONT * txFontExist (const char name[])`

Ищет шрифт по его названию.

2.5. Рисование в памяти (на "виртуальном холсте") и загрузка изображений

- `HDC txCreateCompatibleDC (double sizeX, double sizeY, HBITMAP bitmap=NULL, RGBQUAD **pixels=NULL)`

Создаёт дополнительный холст (контекст рисования, `Device Context, DC`) в памяти.

- `HDC txCreateDIBSection (double sizeX, double sizeY, RGBQUAD **pixels=NULL)`

Создаёт аппаратно-независимый дополнительный холст (контекст рисования, `Device Context, DC`) в памяти с возможностью прямого доступа к нему как к массиву.

- `HDC txLoadImage (const char filename[], int sizeX=0, int sizeY=0, unsigned imageFlags=IMAGE_BITMAP, unsigned loadFlags=LR_LOADFROMFILE)`

Загружает из файла изображение в формате BMP. Делает это довольно медленно.

- `bool txDeleteDC (HDC dc)`

Уничтожает холст (контекст рисования, `DC`) в памяти.

- `bool txBitBlt (HDC destImage, double xDest, double yDest, double width, double height, HDC sourceImage, double xSource=0, double ySource=0, unsigned operation=SRCCOPY)`

Копирует изображение с одного холста (контекста рисования, `DC`) на другой.

- `bool txBitBlt (double xDest, double yDest, HDC sourceImage, double xSource=0, double ySource=0)`

Копирует изображение на экран.

- `bool txTransparentBlt (HDC destImage, double xDest, double yDest, double width, double height, HDC sourceImage, double xSource=0, double ySource=0, COLORREF transColor=TX_BLACK)`

Копирует изображение с одного холста (контекста рисования, `DC`) на другой с учётом прозрачности.

- `bool txTransparentBlt (double xDest, double yDest, HDC sourceImage, COLORREF transColor=TX_BLACK, double xSource=0, double ySource=0)`

Копирует изображение на экран с учётом прозрачности.

- `bool txAlphaBlend (HDC destImage, double xDest, double yDest, double width, double height, HDC sourceImage, double xSource=0, double ySource=0, double al-pha=1.0)`

Копирует изображение с одного холста (контекста рисования, DC) на другой с учётом полупрозрачности.

- `bool txAlphaBlend (double xDest, double yDest, HDC sourceImage, double xSource=0, double ySource=0, double alpha=1.0)`

Копирует изображение на экран с учётом полупрозрачности.

- `HDC txUseAlpha (HDC image)`

Пересчитывает цвета пикселей с учётом прозрачности (переводит цвета в формат Premultiplied Alpha).

- `bool txSaveImage (const char filename[], HDC dc=txDC())`

Сохраняет в файл изображение в формате BMP.

2.6. Вспомогательные функции

- `double txSleep (double time=0)`

Задерживает выполнение программы на определённое время.

- `int txBegin ()`

Блокирует обновление изображения окна, во избежание мигания.

- `int txEnd ()`

Разблокирует обновление окна, заблокированное функцией `txBegin()`.

- `void txRedrawWindow ()`

Обновляет изображение в окне **TXLib** вручную.

- `bool txDestroyWindow (HWND wnd=txWindow())`

Уничтожает окно.

- `double txQueryPerformance ()`

Оценивает скорость работы компьютера.

- `double txGetFPS (int minFrames=txFramesToAverage)`

Выдает количество кадров (вызовов этой функции) в секунду.

- `int txUpdateWindow (int update=true)`

Разрешает или запрещает автоматическое обновление изображения в окне.

- `bool txSelectObject (HGDIOBJ obj, HDC dc=txDC())`

Устанавливает текущий активный объект GDI.

- `bool txIDontWantToHaveAPauseAfterMyProgramBeforeTheWindowWillClose_AndIWillNotBeAskingWhereIsMyPicture ()`

Делает нечто иногда удобное. *См. название функции.*

2.7. Функции консоли

- `unsigned txSetConsoleAttr (unsigned colors=0x07)`

Устанавливает цветовые атрибуты консоли.

- `unsigned txGetConsoleAttr ()`

Возвращает текущие цветовые атрибуты консоли.

- `bool txClearConsole ()`

Стирает текст консоли.

- `POINT txSetConsoleCursorPos (double x, double y)`

Устанавливает позицию мигающего курсора консоли.

- `POINT txGetConsoleCursorPos ()`

Возвращает позицию мигающего курсора консоли.

- `POINT txGetConsoleExtent ()`

Возвращает размер консоли.

- `POINT txGetConsoleFontSize ()`

Возвращает размеры шрифта консоли.

- `bool txTextCursor (bool blink=true)`

Запрещает или разрешает рисование мигающего курсора в окне.

2.8. Функции

2.8.1. `HWND txCreateWindow (double sizeX, double sizeY, bool centered = true)`

Создание окна рисования

Аргументы:

sizeX	Размер окна по горизонтали (в пикселях).
sizeY	Размер окна по вертикали (в пикселях).
centered	Центрирование окна на дисплее. Необязательно. Если не указано, то окно центрируется.

Возвращает:

Дескриптор (системный номер) окна **TXLib**. Если окно не создано, возвращается `NULL`.

Предупреждения:

- *Если используется многофайловый проект (с раздельной компиляцией), то графические функции* **TXLib***, вызванные из файла проекта, будут работать только с тем окном, которое создано в этом же файле проекта. Если проект состоит, скажем, из файлов* `main.cpp` *и* `game.cpp`*, и в файле* `main.cpp` *создаётся графическое окно, то функции из* `game.cpp` *не смогут рисовать в нём. (Однако* `game.cpp` *сможет создать своё собственное окно.)*
- *Если такой программе нужно одно окно, то в проект следует включить файл, ответственный за рисование, скажем,* `graphics.cpp`*, и выводить графику только через функции этого файла. Такой файл (или библиотеку) в больших проектах часто называют графическим движком.*
- *То же касается и использования* **TXLib** *в* **DLL***.*

Заметки:

- Вспомогательные окна могут создаваться по одному на каждый файл многофайлового проекта или загруженную **DLL**. Для закрытия вспомогательных окон используется `txDestroyWindow()`. Для закрытия главного надо выйти из `main()`.
- Одновременное создание нескольких окон не потокобезопасно ***(not thread-safe)***.
- Многооконная программа на **TXLib** тормозит, да и однооконная тоже не отличается высокой скоростью. Чтобы избавиться от этого, бросьте **TXLib** и используйте другие оконные библиотеки: **Qt, wxWidgets, GTK+** и т.д., или библиотеки, специально разработанные для создания игр: **SFML** (она простая), **SDL** и другие. Вы можете написать и свою соб-

ственную оконную библиотеку, основанную на **OpenGL** и получится **SFML** или **SDL**:), или **DirectX**. Помните, что цель **TXLib** — облегчить первые шаги, но потом стать ненужной.

- Устанавливаются параметры рисования по умолчанию, *см. функцию* `txSetDefaults()`.

См. также:

`txOK()`, `txWindow()`, `txDC()`, `txVideoMemory()`, `_txWindowStyle`, `_txConsoleMode`, `_txConsoleFont`, `_txCursorBlinkInterval`, `_txWindowUpdateInterval`, `_TX_NOINIT`, `_TX_ALLOW_TRACE`, `TX_TRACE`

Примеры использования:

```
txCreateWindow (800, 600);              //Окно 800x600, центрировано
txCreateWindow (1024, 768, false);      //Окно 1024x768, не центрировано
```

2.8.2. HDC& txDC ()

Возвращает холст (дескриптор контекста рисования, HDC) окна рисования **TXLib**.

Возвращает:

Дескриптор (системный номер, `handler`) контекста рисования (`device context, DC`) холста **TXLib** (TXLib HDC).

Заметки:

- Возвращаемый дескриптор — не оконный контекст рисования окна **TXLib**. **TXLib** реализует двойную буферизацию. Все рисовательные действия происходят со скрытым HDC, находящемся в памяти, и его содержимое периодически автоматически копируется на экран. Это иногда приводит к мерцанию. Автоматическое копирование можно выключить функцией `txBegin()` и обратно включить функцией `txEnd()`, в этом случае содержимое окна можно перерисовать функциями `txRedraw-Window()` или `txSleep()`.

- Дополнительную информацию об автоматическом *обновлении см.* в функциях `txBegin()`, `txEnd()`, `txUpdateWindow()`, `txRedrawWindow()` и `txSleep()`.
- Этот HDC возвращается в виде ссылки, что позволяет подменить его. Тогда **TXLib** будет периодически копировать на экран изображение уже из вашего буфера. Перед подменой надо сохранить старый дескриптор или освободить его с помощью `txDeleteDC()`. Во время подмены рисование должно быть заблокировано с помощью `txLock()` и после разблокировано с помощью `txUnlock()`.

См. также:

`txWindow()`, `txBegin()`, `txEnd()`, `txLock()`, `txUnlock()`, `txGDI()`

Примеры использования:

```
txBitBlt (txDC(), 0, 0, 100, 100, txDC(), 0, 0);
txBitBlt (txDC(), 100, 0, 100, 100, txDC(), 0, 0);
txBitBlt (txDC(), 0, 100, 100, 100, txDC(), 0, 0);
txBitBlt (txDC(), 100, 100, 100, 100, txDC(), 0, 0);
```

2.8.3. RGBQUAD* txVideoMemory ()

Возвращает буфер памяти, связанный с холстом (HDC) **TXLib**.

Возвращает:

- Указатель на массив структур `RGBQUAD`, — буфера памяти, связанного с HDC холста **TXLib**.
- Прямой доступ к буферу памяти HDC позволяет работать с ним с очень высокой

скоростью. Правда, рисовать можно только отдельные пиксели. Это полезно, в основном, если вы пишете собственный графический рендер *(так напишите же его!)*.

- Буфер памяти HDC — двумерный массив, размеры которого соответствуют ширине и высоте холста (HDC). Но он возвращается как указатель на одномерный массив, поэтому двумерную адресацию к нему надо вести вручную. Кроме того, "*Y-ось*" этого массива направлена вверх, а не вниз, как в окне **TXLib**. Поэтому для нужного пикселя его смещение от начала массива нужно рассчитывать с помощью формулы `x + (-y + sizeY) * sizeX`.

Предупреждения:

Будьте осторожны, не выходите за границы массива, последствия будут непредсказуемыми.

Заметки:

- Во время работы с буфером автоматическое обновление окна **TXLib** должно быть заблокировано с помощью `txLock()` и после разблокировано с помощью `txUnlock()`.
- **TXLib** — не оконный контекст рисования окна **TXLib**. **TXLib** реализует двойную буферизацию. Все рисовательные действия происходят со скрытым HDC, находящемся в памяти (его возвращает `txGetDC()`), и его содержимое периодически автоматически копируется на экран. Это иногда приводит к мерцанию. Автоматическое копирование можно выключить функцией `txBegin()` и обратно включить функцией `txEnd()`, в этом случае содержимое окна можно перерисовать функциями `txRedrawWindow()` или `txSleep()`.
- Дополнительную информацию об автоматическом *обновлении см.* в функциях `txBegin()`, `txEnd()`, `txUpdateWindow()`, `txRedrawWindow()` и `txSleep()`.

См. также:

txCreateDIBSection(), txDC(), txWindow(), txBegin(), txEnd(), txLock(), txUnlock(), txGDI()

Примеры использования:

Пример см. в файле PhongDemo.cpp из папки TX\Examples\Demo.

Также см. пример в помощи по функции txCreateDIBSection().

2.8.4. bool txSetDefaults (HDC dc = txDC())

Установка параметров рисования по умолчанию.

Аргументы:

dc	Дескриптор контекста рисования (холста) для установки параметров. Необязателен.

Возвращает:

Если операция была успешна — true, иначе — false.

Параметры по умолчанию:

- Линии — цвет белый (TX_WHITE), толщина 1
- Заливка — цвет белый (TX_WHITE)
- Шрифт — Системный, цвет белый (TX_WHITE)
- Растровая операция — копирование цвета (R2_COPYPEN)

См. также:

txSetColor(), txGetColor(), txSetFillColor(), txGetFillColor(), txColors, RGB(), txSelectFont()

Примеры использования:

```
txSetDefaults();
```

2.8.5. bool txOK ()

Проверка правильности работы библиотеки

Возвращает:

Состояние библиотеки: `true` — библиотека в порядке, `false` — не в порядке.

«Библиотека не в порядке» означает, что её внутренние данные неверны. Самая простая причина — не открыто окно, однако могут быть и другие проблемы.

См. также:

`txCreateWindow()`

Примеры использования:

```
txCreateWindow (800, 600);
if (!txOK())
      {
      txMessageBox ("Не смогла создать окно", "Извините", MB_ICONSTOP);
      return;
      }
```

2.8.6. POINT txGetExtent (HDC dc = txDC())

Возвращает размер окна, картинки или холста в виде структуры `POINT`.

Аргументы:

dc	Дескриптор контекста рисования (холста) с загруженной или созданной картинкой, размеры которого нужно определить. Необязателен. Если не указан или равен `txDC()`, возвращаются размеры окна **TXLib**.

Возвращает:

Размер окна, картинки или холста в виде структуры `POINT`, содержащей компоненты x и y.

Заметки:

Если окно не создано, возвращается размер экрана.

См. также:

`txGetExtentX()`, `txGetExtentY()`

Примеры использования:

```
txCreateWindow (800, 600);
HDC image = txLoadImage ("TX Application_resize.bmp");
POINT center = { txGetExtentX() / 2, txGetExtentY() / 2 };
POINT size = txGetExtent (image);
txBitBlt (center.x, center.y, image); // This is krivo
```

```
txBitBlt (center.x - size.x/2, center.y - size.y/2, image); // This is centered
txSetColor (TX_WHITE, 3);
txSetFillColor (TX_TRANSPARENT);
txCircle (center.x, center.y, hypot (size.x, size.y) / 2);
txCircle (center.x, center.y, 10);
txDeleteDC (image);
```

2.8.7. int txGetExtentX (HDC dc = txDC())

Возвращает ширину окна или холста.

Аргументы:

dc	Дескриптор контекста рисования (холста), ширина которого возвращается. Необязателен. Если не указан или равен txDC(), возвращаются ширина окна **TXLib**.

Возвращает:

Ширина окна рисования.

Заметки:

Если окно не создано, возвращается ширина экрана.

См. также:

txGetExtent(), txGetExtentY()

Примеры использования:

```
txSetTextAlign (TA_CENTER);
txTextOut (txGetExtentX() / 2, 100, "Oh, oh, you're in the [army]middle now");
```

2.8.8. int txGetExtentY (HDC dc = txDC())

Возвращает высоту окна или холста.

Аргументы:

dc	Дескриптор контекста рисования (холста), высота которого возвращается. Необязателен. Если не указан или равен `txDC()`, возвращается высота окна **TXLib**.

Возвращает:

Высота окна рисования.

Заметки:

Если окно не создано, возвращается высота экрана.

См. также:

`txGetExtent()`, `txGetExtentX()`

Примеры использования:

```
void DrawHouse (int height);
...
DrawHouse (txGetExtentY() / 2);
```

2.8.9. HWND txWindow ()

Возвращает дескриптор окна рисования

Возвращает:

Дескриптор (системный номер, `handler`) окна холста.

См. также:

`txDC()`, `txVideoMemory()`, `txLock()`, `txUnlock()`, `txGDI()`

Примеры использования:

```
SetWindowText (txWindow(), "Новые заголовки — теперь и в ваших окнах!");
txMessageBox ("Распишитесь", "Получите", MB_ICONINFORMATION);
```

2.8.10. COLORREF RGB (int red, int green, int blue)

Создаёт (смешивает) цвет из трёх базовых цветов (компонент).

Аргументы:

red	Количество красного цвета в интервале [0; 255].
green	Количество зеленого цвета в интервале [0; 255].
blue	Количество синего цвета в интервале [0; 255]

Возвращает:

Созданный цвет в формате `COLORREF`.

См. также:

txSetColor(), txGetColor(), txSetFillColor(), txGetFillColor(), txColors, RGB(), txExtractColor(), txRGB2HSL(), txHSL2RGB()

Примеры использования:

```
txSetColor (RGB (255, 128, 0));                 // Красный + половина зеленого = оранжевый

int red = 20, green = 200, blue = 20;
COLORREF color = RGB (red, green, blue);
txSetFillColor (color);

const COLORREF SKY_COLOR = RGB (0, 128, 255);     // Создаем константу для цвета
...
txSetFillColor (SKY_COLOR);                       // Используем ее
```

2.8.11. HPEN txSetColor (COLORREF color, double thickness = 1, HDC dc = txDC())

Устанавливает текущий цвет и толщину линий, цвет текста.

Аргументы:

color	Цвет линий и текста, см. txColors, RGB().
thickness	Толщина линий. Необязательна. Если не указана, то 1 пиксель.
dc	Дескриптор контекста рисования (холста) для установки цвета. Необязателен.

Возвращает:

Перо, созданное при установке цвета. При ошибке возвращается `NULL`.

См. также:

`txSetFillColor()`, `txGetColor()`, `txGetFillColor()`, `txColors`, `RGB()`

Примеры использования:

```
txSetColor (TX_YELLOW);              // Цвет линий будет желтым
txSetColor (RGB (255, 128, 0), 5);   // Оранжевые линии толщиной 5 пикселей

txSetColor (RGB (255, 255, 0));      // Желт.= Красн.+ зел. (др.способ указ.цв.)
txSetColor (RGB (255, 128, 64));     // Нечто оранжевое
```

2.8.12. COLORREF txGetColor (HDC dc = txDC())

Возвращает текущий цвет линий и текста.

Аргументы:

dc	Дескриптор контекста рисования (холста) для возврата цвета. Необязателен.

Возвращает:

Текущий цвет линий и текста, *см.* `txColors`, `RGB()`.

См. также:

`txSetColor()`, `txSetFillColor()`, `txGetFillColor()`, `txColors`, `RGB()`

Примеры использования:

```
COLORREF color = txGetColor();
```

2.8.13. HBRUSH txSetFillColor (COLORREF color, HDC dc = txDC())

Устанавливает текущий цвет заполнения фигур.

Аргументы:

color	Цвет заполнения, см. txColors, RGB().
dc	Дескриптор контекста рисования (холста) для установки цвета. Необязателен.

Возвращает:

Кисть, созданная при установке цвета. При ошибке возвращается `NULL`.

См. также:

`txSetColor()`, `txGetFillColor()`, `txGetColor()`, `txColors`, `RGB()`

Примеры использования:

```
txSetFillColor (TX_LIGHTGREEN);
txSetFillColor (RGB (255, 128, 0));
```

2.8.14. COLORREF txGetFillColor (HDC dc = txDC())

Возвращает текущий цвет заполнения фигур.

Аргументы:

dc	Дескриптор контекста рисования (холста) для возврата цвета. Необязателен.

Возвращает:

Текущий цвет заполнения фигур, *см.* `txColors, RGB()`.

См. также:

`txSetFillColor(), txSetColor(), txGetColor(), txColors, RGB()`

Примеры использования:

```
COLORREF color = txGetFillColor();
```

2.8.15. unsigned txExtractColor (COLORREF color, COLORREF component)

Извлекает цветовую компоненту (цветовой канал) из смешанного цвета.

Аргументы:

color	Смешанный цвет.
component	Извлекаемая компонента, *см.* `txColors`.

Возвращает:

Цветовая компонента, *см.* `txColors`.

См. также:

`txSetColor(), txGetColor(), txSetFillColor(), txGetFillColor(), txColors, RGB(), txExtractColor(), txRGB2HSL(), txHSL2RGB()`

Примеры использования:

```
int red = txExtractColor (color, TX_RED);
int lightness = txExtractColor (TX_BLUE, TX_LIGHTNESS);
```

Другие примеры *см. в функциях* `AppearText(), AppearEarth()` ***Примера 5 («Циклы»)***.

2.8.16. COLORREF txRGB2HSL (COLORREF rgbColor)

Преобразует цвет из формата RGB в формат HSL.

Аргументы:

rgbColor	Преобразуемый цвет в формате ЕГЭ RGB.

Возвращает:

Созданный цвет в виде COLORREF.

- Формат RGB определяется как

Красная компонента цвета (Red), от **0** до **255**.

Зелёная компонента цвета (Green), от **0** до **255**.

Синяя компонента цвета (Blue), от **0** до **255**.

- Формат HSL определяется как

Цветовой тон (Hue), от **0** до **255** *(не до* **360**).

Насыщенность (Saturation), от **0** до **255**.

Светлота (Lightness), от **0** до **255**.

См. также:

txSetColor(), txGetColor(), txSetFillColor(), txGetFillColor(), txColors, RGB(), txExtractColor(), txRGB2HSL(), txHSL2RGB()

Примеры использования:

```
COLORREF hslColor = txRGB2HSL (TX_RED);
```

2.8.17. COLORREF txHSL2RGB (COLORREF hslColor)

Преобразует цвет из формата HSL в формат RGB.

Аргументы:

hslColor	Преобразуемый цвет в формате HSL.

Возвращает:

Созданный цвет в виде COLORREF.

- Формат RGB определяется как

Красная компонента цвета (Red), от **0** до **255**.

Зелёная компонента цвета (Green), от **0** до **255**.

Синяя компонента цвета (Blue), от **0** до **255**.

- Формат HSL определяется как

Цветовой тон (Hue), от **0** до **255** *(не до* **360***)*.

Насыщенность (Saturation), от **0** до **255**.

Светлота (Lightness), от **0** до **255**.

См. также:

txSetColor(), txGetColor(), txSetFillColor(), txGetFillColor(), txColors, RGB(), txExtractColor(), txRGB2HSL(), txHSL2RGB()

Примеры использования:

```
int hue = 10, saturation = 128, lightness = 128;
COLORREF hslColor = RGB (hue, saturation, lightness);
txSetColor (txHSL2RGB (hslColor));
```

2.8.18. bool txClear (HDC dc = txDC())

Стирает холст текущим цветом заполнения.

Аргументы:

dc	Дескриптор контекста рисования (холста) для стирания. Необязателен.

Возвращает:

Если операция была успешна – `true`, иначе – `false`.

См. также:

`txSetFillColor()`, `txGetFillColor()`, `txColors`, `RGB()`

Примеры использования:

```
txSetFillColor (TX_BLUE); // Кто-то хотел синий фон?
txClear();
```

2.8.19. bool txSetPixel (double x, double y, COLORREF color, HDC dc = txDC())

Рисует пиксель (точку на экране).

Аргументы:

x	X-координата точки.
y	Y-координата точки.
color	Цвет точки, *см.* `txColors, RGB()`.
dc	Дескриптор контекста рисования (холста) для рисования. Необязателен.

Возвращает:

Если операция была успешна – `true`, иначе – `false`.

См. также:

`txGetPixel(), txColors, RGB(), txVideoMemory()`

Примеры использования:

```
txSetPixel (100, 100, TX_LIGHTRED);    // Красная точка!
                                       // http://www.google.ru/search?q=коты+и+красная+точка
txSetPixel (100, 100, RGB (255, 128, 0));
```

2.8.20. COLORREF txGetPixel (double x, double y, HDC dc = txDC())

Возвращает текущий цвет точки (пикселя) на экране.

Аргументы:

x	X-координата точки.
y	Y-координата точки.
dc	Дескриптор контекста рисования (холста) для возврата цвета. Необязателен.

Возвращает:

Текущий цвет пикселя, *см.* `txColors, RGB()`.

См. также:

`txSetPixel(), txColors, RGB(), txVideoMemory()`

Примеры использования:

```
COLORREF color = txGetPixel (100, 200);
if (txGetPixel (x, y) == TX_RED)
CarCrash (x, y);                        // Mess with the red — die like the rest
```

2.8.21. bool txLine (double x0, double y0, double x1, double y1, HDC dc = txDC())

Рисует линию.

Аргументы:

x0	X-координата начальной точки.
y0	Y-координата начальной точки.
x1	X-координата конечной точки.
y1	Y-координата конечной точки.
dc	Дескриптор контекста рисования (холста) для рисования линии. Необязателен.

Возвращает:

- Если операция была успешна – `true`, иначе – `false`.
- Цвет и толщина линии задаётся функцией `txSetColor()`.

См. также:

`txSetColor()`, `txGetColor()`, `txSetFillColor()`, `txGetFillColor()`, `txColors`, `RGB()`, `txLine()`, `txRectangle()`, `txPolygon()`, `txEllipse()`, `txCircle()`, `txArc()`, `txPie()`, `txChord()`

Примеры использования:

```
txLine (10, 50, 100, 500);        // Правда бедный примерчик, да?
```

2.8.22. bool txRectangle (doublex0, double y0, double x1, double y1, HDC dc = txDC())

Рисует прямоугольник.

Аргументы:

x0	X-координата верхнего левого угла.
y0	Y-координата верхнего левого угла.
x1	X-координата нижнего правого угла.
y1	Y-координата нижнего правого угла.
dc	Дескриптор контекста рисования (холста) для рисования прямоугольника. Необязателен.

Возвращает:

- Если операция была успешна — `true`, иначе — `false`.

- Цвет и толщина линий задаётся функцией `txSetColor()`, цвет заполнения — `txSetFillColor()`.

См. также:

`txSetColor()`, `txGetColor()`, `txSetFillColor()`, `txGetFillColor()`, `txColors`, `RGB()`, `txLine()`, `txRectangle()`, `txPolygon()`, `txEllipse()`, `txCircle()`, `txArc()`, `txPie()`, `txChord()`

Примеры использования:

```
txRectangle (100, 200, 400, 500);

Win32::RoundRect (txDC(), 100, 200, 400, 500, 30, 30);
// И такое есть. Погуглите "RoundRect function".
```

2.8.23. bool txPolygon (const POINT points[], int numPoints, HDC dc = txDC())

Рисует ломаную линию или многоугольник.

Аргументы:

points	Массив структур `POINT` с координатами точек.
numPoints	Количество точек в массиве.
dc	Дескриптор контекста рисования (холста) для рисования. Необязателен.

Возвращает:

Если операция была успешна — `true`, иначе — `false`.

Цвет и толщина линий задаётся функцией `txSetColor()`, цвет заполнения — `txSetFillColor()`. Если нужно нарисовать ломаную линию, а не многоугольник, используйте прозрачный цвет заполнения `(TX_TRANSPARENT)`.

См. также:

txSetColor(), txGetColor(), txSetFillColor(), txGetFillColor(), txColors, RGB(), txLine(), txRectangle(), txPolygon(), txEllipse(), txCircle(), txArc(), txPie(), txChord()

Примеры использования:

```
POINT star[5] = {{150, 300}, {200, 100}, {250, 300}, {100, 200}, {300, 200}};
txPolygon (star, 5); // Я кривая звездочка
```

2.8.24. bool txEllipse (double x0, double y0, double x1, double y1, HDC dc = txDC())

Рисует эллипс.

Аргументы:

x0	X-координата верхнего левого угла прямоугольника, описанного вокруг эллипса.
y0	Y-координата верхнего левого угла описанного прямоугольника.
x1	X-координата нижнего правого угла описанного прямоугольника.
y1	Y-координата нижнего правого угла описанного прямоугольника.
dc	Дескриптор контекста рисования (холста) для рисования. Необязателен.

Возвращает:

Если операция была успешна — **true**, иначе — **false**.

Цвет и толщина линий задаётся функцией **txSetColor()**, цвет заполнения — **txSetFillColor()**.

См. также:

txSetColor(), txGetColor(), txSetFillColor(), txGetFillColor(), txColors, RGB(), txLine(), txRectangle(), txPolygon(), txEllipse(), txCircle(), txArc(), txPie(), txChord()

Примеры использования:

```
txEllipse (100, 100, 300, 200);
```

2.8.25. bool txCircle (double x, double y, double r)

Рисует окружность или круг.

Аргументы:

x	X-координата центра.
y	Y-координата центра.
r	Радиус.

Возвращает:

Если операция была успешна — true, иначе — false.

Цвет и толщина линий задаётся функцией txSetColor(), цвет заполнения — txSetFillColor().

См. также:

txSetColor(), txGetColor(), txSetFillColor(), txGetFillColor(), txColors, RGB(), txLine(), txRectangle(), txPolygon(), txEllipse(), txCircle(), txArc(), txPie(), txChord()

Примеры использования:

```
txCircle (100, 100, 10);
```

2.8.26. bool txArc (double x0, double y0, double x1, double y1, double startAngle, double totalAngle, HDC dc = txDC())

Рисует дугу эллипса.

Аргументы:

x0	X-координата верхнего левого угла прямоугольника, описанного вокруг эллипса, содержащего дугу *(см.* `txEllipse`*)*.
y0	Y-координата верхнего левого угла прямоугольника.
x1	X-координата нижнего правого угла прямоугольника.
y1	Y-координата нижнего правого угла прямоугольника
startAngle	Угол между направлением оси ***OX*** и началом дуги (в градусах).
totalAngle	Величина дуги (в градусах).
dc	Дескриптор контекста рисования (холста) для рисования. Необязателен.

Возвращает:

Если операция была успешна — `true`, иначе — `false`.

Цвет и толщина линий задаётся функцией `txSetColor()`, цвет заполнения — `txSetFillColor()`.

См. также:

`txSetColor()`, `txGetColor()`, `txSetFillColor()`, `txGetFillColor()`, `txColors`, `RGB()`, `txLine()`, `txRectangle()`, `txPolygon()`, `txEllipse()`, `txCircle()`, `txArc()`, `txPie()`, `txChord()`

Примеры использования:

```
txArc (100, 100, 300, 200, 45, 270);
```

2.8.27. bool txPie(double x0, double y0, double x1, double y1, double startAngle, double totalAngle, HDC dc = txDC())

Рисует сектор эллипса.

Аргументы:

x0	X-координата верхнего левого угла прямоугольника, описанного вокруг эллипса, содержащего дугу *(см.* `txEllipse`*)*.
y0	Y-координата верхнего левого угла прямоугольника.
x1	X-координата нижнего правого угла прямоугольника.
y1	Y-координата нижнего правого угла прямоугольника.
startAngle	Угол между направлением оси ***OX*** и началом сектора (в градусах).
totalAngle	Величина сектора (в градусах).
dc	Дескриптор контекста рисования (холста) для рисования. Необязателен.

Возвращает:

- Если операция была успешна — `true`, иначе — `false`.
- Цвет и толщина линий задаётся функцией `txSetColor()`, цвет заполнения — `txSetFillColor()`.

См. также:

`txSetColor()`, `txGetColor()`, `txSetFillColor()`, `txGetFillColor()`, `txColors`, `RGB()`, `txLine()`, `txRectangle()`, `txPolygon()`, `txEllipse()`, `txCircle()`, `txArc()`, `txPie()`, `txChord()`

Примеры использования:

```
txPie (100, 100, 300, 200, 0, 180); // Enter Pacman
```

2.8.28. bool txChord (double x0, double y0, double x1, double y1, double startAngle, double totalAngle, HDC dc = txDC())

Рисует хорду эллипса.

Аргументы:

x0	X-координата верхнего левого угла прямоугольника, описанного вокруг эллипса, содержащего дугу *(см.* `txEllipse`*)*.
y0	Y-координата верхнего левого угла прямоугольника.
x1	X-координата нижнего правого угла прямоугольника.
y1	Y-координата нижнего правого угла прямоугольника.
startAngle	Угол между направлением оси ***OX*** и началом хорды (в градусах).
totalAngle	Величина хорды (в градусах).
dc	Дескриптор контекста рисования (холста) для рисования. Необязателен.

Возвращает:

- Если операция была успешна — `true`, иначе — `false`.
- Цвет и толщина линий задаётся функцией `txSetColor()`, цвет заполнения — `txSetFillColor()`.

См. также:

`txSetColor()`, `txGetColor()`, `txSetFillColor()`, `txGetFillColor()`, `txColors`, `RGB()`, `txLine()`, `txRectangle()`, `txPolygon()`, `txEllipse()`, `txCircle()`, `txArc()`, `txPie()`, `txChord()`

Примеры использования:

```
txChord (100, 100, 300, 200, 45, 270);
```

2.8.29. bool txFloodFill (double x, double y, COLORREF color = TX_TRANSPARENT, DWORD mode = FLOODFILLSURFACE, HDC dc = txDC())

Заливает произвольный контур текущим цветом заполнения.

Аргументы:

x	X-координата точки начала заливки.
y	Y-координата точки начала заливки.
color	Цвет заливаемой области. Необязателен. Если не указан, то используется TX_TRANSPARENT — автоопределение цвета.
mode	Режим заливки. Необязателен. Если не указан, то используется FLOODFILLSURFACE — заливка однородного фона.
dc	Дескриптор контекста рисования (холста) для рисования. Необязателен.

Возвращает:

Если операция была успешна — true, иначе — false.

Предупреждения:

Цвет заполнения задаётся функцией txSetFillColor(). *Не рекомендуется для применения, так как работает довольно медленно. Лучше Используйте* txPolygon().

Режимы заливки:

FLOODFILLSURFACE	Заливать область, указанную цветом color.
FLOODFILLBORDER	Заливать до границы, указанной цветом color.

См. также:

txSetFillColor(), txGetFillColor(), txColors, RGB(), txLine(), txRectangle(), txPolygon(), txEllipse(), txCircle(), txArc(), txPie(), txChord()

Примеры использования:

```
txSetFillColor (TX_PINK);
txLine (100, 200, 150, 100);
txLine (150, 100, 200, 200);
txLine (200, 200, 100, 200);
txFloodFill (150, 150);
```

2.8.30. bool txTriangle (double x1, double y1, double x2, double y2, double x3, double y3)

Функция, которая должна бы рисовать треугольник.

Аргументы:

x1	X-координата 1-й точки.
y1	Y-координата 1-й точки.
x2	X-координата 2-й точки.
y2	Y-координата 2-й точки.
x3	X-координата 3-й точки.
y3	Y-координата 3-й точки.

Возвращает:

Если операция была бы успешна — `true`, иначе — `false`.

См. также:

`txSetFillColor()`, `txGetFillColor()`, `txColors`, `RGB()`, `txLine()`, `txRectangle()`, `txPolygon()`, `txEllipse()`, `txCircle()`, `txArc()`, `txPie()`, `txChord()`

См. также:

Пример с функциями с параметрами

2.8.31. void txDrawMan (int x, int y, int sizeX, int sizeY, COLORREF color, double handL, double handR, double twist, double head, double eyes, double wink, double crazy, double smile, double hair, double wind)

Рисует человечка.

Это пример функции, которую Вы могли бы написать и сами.

Аргументы:

x	X-координата человечка.
y	Y-координата человечка.
sizeX	Ширина человечка.
sizeY	Высота человечка (также определяет размер головы).
color	Цвет человечка.
handL	Высота подъёма левой руки (относительно высоты человечка).
handR	Высота подъёма правой руки (относительно высоты человечка).

twist	Смещение спины (относительно ширины человечка).
head	Высота подъёма головы (относительно высоты человечка).
eyes	Величина глаз (относительно размера головы).
wink	Моргание глаз (**0** — оба открыты, **-1** — закрыт левый, **+1** — закрыт правый).
crazy	Смещение глаз по вертикали (относительно размера головы).
smile	Улыбка (относительно размера головы).
hair	Длина волос (относительно размера головы).
wind	Ветер, развевающий волосы (относительно размера головы).

См. также:

`txSetFillColor()`, `txColors`, `RGB()`, `txLine()`, `txCircle()`

Примеры использования:

```
txCreateWindow (800, 600);
//--------+---+---+-----+----+-------+----+-----+----+----+---+----+----+----+----+----
//        | x | y |sizeX|sizeY| color |handL|handR|twist|head|eyes|wink|crazy|smile|hair|wind
//--------+---+---+-----+----+-------+----+-----+----+----+---+----+----+----+----+----
//        |   |   |     |    |       |    |     |    |    |   |    |    |    |    |
txDrawMan (125, 250, 200, 200, TX_WHITE,    0,  0,  0,  0,  0.8,  0,  0,  1.0,  0,  0);
txDrawMan (325, 250, 100, 200, TX_YELLOW,   0,  0,  0,  0,  0.8,  0,  0, -1.0,  2,  0);
txDrawMan (525, 250, 200, 100, TX_ORANGE,   0,  0,  0,  0,  1.0,  0, -1,  0.3,  1,  0);
txDrawMan (725, 250, 100, 100, TX_LIGHTRED, 0,  0,  0,  0,  1.0,  0,  1, -0.3,  3,  0);

txDrawMan (125, 550, 200, 200, TX_WHITE,    0.3, 0.3, 0, 0, 0.8, -1, 1,   0.5, 2, -1);
txDrawMan (325, 550, 100, 200, TX_YELLOW,   -0.5, -0.5, 0, 0.1, 0.8,   1, 0,   -0.5, 3,  5);
txDrawMan (525, 550, 200, 100, TX_ORANGE,  -0.5, 0.3,  0.2, 0, 0.8,  -1, 1,   0.0, 10, -5);
txDrawMan (725, 550, 100, 100, TX_LIGHTRED,   0.3, -0.5, -0.4, 0, 0.8,   1, -1,  0.0,  1,   1);
```

2.8.32. bool txTextOut (double x, double y, const char text[], HDC dc = txDC())

Рисует текст.

Аргументы:

x	X-координата начальной точки текста.
y	Y-координата начальной точки текста.

text	Текстовая строка.
dc	Дескриптор контекста рисования (холста) для рисования. Необязателен.

Возвращает:

Если операция была успешна — `true`, иначе — `false`.

Цвет текста задается функцией `txSetColor()`, выравнивание (влево/вправо/по центру) — `txSetTextAlign()`.

См. также:

`txSetColor()`, `txGetColor()`, `txSetFillColor()`, `txGetFillColor()`, `txColors`, `RGB()`, `txSelectFont()`, `txSetTextAlign()`, `txGetTextExtent()`, `txGetTextExtentX()`, `txGetTextExtentY()`

Примеры использования:

```
txTextOut (100, 100, "Здесь могла бы быть Ваша реклама.");
```

2.8.33. bool txDrawText (double x0, double y0, double x1, double y1, const char text[], unsigned format = DT_CENTER|DT_VCENTER|DT_WORDBREAK|DT_WORD_ELLIPSIS, HDC dc = txDC())

Рисует текст, размещённый в прямоугольной области.

Аргументы:

x0	X-координата верхнего левого угла области.
y0	Y-координата верхнего левого угла области.
x1	X-координата нижнего правого угла области.
y1	Y-координата нижнего правого угла области.

text	Текстовая строка.
format	Флаги форматирования текста. Необязательны. Если не указаны, то используется: центрирование, перенос по словам и добавление многоточия, если надпись не умещается в область.
dc	Дескриптор контекста рисования (холста) для рисования. Необязателен.

Возвращает:

Если операция была успешна — `true`, иначе — `false`.

Цвет текста задаётся функцией `txSetColor()`, выравнивание (влево/вправо/по центру) — `txSetTextAlign()`.

Заметки:

- Не выводит ничего, если координаты идут в неверном порядке (если x0 > x1 или y0 > y1).
- Флаги форматирования текста *см. в* `MSDN (http://msdn.com)`, искать `"DrawText Function (Windows)": http://msdn.microsoft.com/enus/library/dd162498%28VS.85%29.aspx`.
- Автоматический перенос текста на несколько строк включается, если текст состоит из нескольких строк (есть хотя бы один символ новой строки `\n`).
- Если надо отформатировать текст не по центру, а по левому или правому краю, то не забудьте указать остальные флаги форматирования, если они нужны: `DT_VCENTER` (вертикальное центрирование) `|` `DT_WORDBREAK` (перенос по словам) `|` `DT_WORD_ELLIPSIS` (ставить многоточие в конце, если текст не умещается). *См.* значение флагов по умолчанию.
- Вертикальное центрирование работает только для надписей, в которых нет символа новой строки `\n`.

См. также:

txSetColor(), txGetColor(), txSetFillColor(), txGetFillColor(), txColors, RGB(), txTextOut(), txSelectFont(), txGetTextExtent(), txGetTextExtentX(), txGetTextExtentY()

Примеры использования:

```
txSetColor (TX_BLACK);
txSetFillColor (TX_DARKGRAY);   Win32::RoundRect (txDC(), 105, 105, 205, 255, 30, 30);
txSetFillColor (TX_WHITE);      Win32::RoundRect (txDC(), 100, 100, 200, 250, 30, 30);

txSelectFont ("Arial", 20, 0, FW_BOLD);
txDrawText (100, 100, 200, 250, "I hate it when I'm studying"
                                "and a Velociraptor throws bananas on me.\n");
```

2.8.34. HFONT txSelectFont (const char name[], double sizeY, double sizeX=-1, int bold=FW_DONTCARE, bool italic = false, bool underline = false, bool strikeout = false, double angle = 0, HDC dc = txDC())

Выбирает текущий шрифт, его размер и другие атрибуты.

Аргументы:

name	Название шрифта.
sizeY	Высота букв (размер по Y).
sizeX	Ширина букв. Необязательна. Если не указана, то берётся ***1/3*** от высоты.

bold	Жирность шрифта (от 0 до 1000). Необязательна. Если не указана, то берётся обычный шрифт.
italic	Курсив. Необязателен.
underline	Подчёркивание. Необязательно.
strikeout	Перечёркивание. Необязательно.
angle	Угол поворота текста (в градусах). Необязателен.
dc	Дескриптор контекста рисования (холста) для выбора шрифта. Необязателен.

Возвращает:

Выбранный шрифт. Если он не был найден, то устанавливается системный шрифт `Windows (SYSTEM_FIXED_FONT,` *см.* `MSDN)`. Существование шрифта можно проверить функцией `txFontExist()`.

См. также:

`txTextOut(), txDrawText(), txFontExist()`

Примеры использования:

```
txSelectFont       ("Comic Sans MS", 40);
txTextOut    (100, 100, "И здесь могла бы быть Ваша реклама.");
txSelectFont       ("Comic Sans MS", 40, 10, false, true, false, true, 15);
txTextOut    (100, 200, "Но её почему-то нет.");
```

2.8.35. SIZE txGetTextExtent (const char text[], HDC dc = txDC())

Вычисляет размеры текстовой надписи.

Аргументы:

text	Текстовая строка.
dc	Дескриптор контекста рисования (холста), где планируется разместить надпись. Необязателен.

Возвращает:

Размеры надписи в структуре SIZE.

См. также:

txTextOut(), txSelectFont(), txGetTextExtent(), txGetTextExtentX(), txGetTextExtentY()

Примеры использования:

```
SIZE size = txGetTextExtent (text);
txTextOut    (100 + size.cx / 2, 200 + size.cy / 2, text);
```

2.8.36. int txGetTextExtentX (const char text[], HDC dc = txDC())

Вычисляет ширину текстовой надписи.

Аргументы:

text	Текстовая строка.
dc	Дескриптор контекста рисования (холста), где планируется разместить надпись. Необязателен.

Возвращает:

Ширина надписи.

См. также:

`txTextOut()`, `txSelectFont()`, `txGetTextExtent()`, `txGetTextExtentX()`, `txGetTextExtentY()`

Примеры использования:

```
txTextOut (100 + txGetTextExtentX (text) / 2, 200 + txGetTextExtentY (text) / 2, text);
```

2.8.37. `int txGetTextExtentY (const char text[], HDC dc = txDC())`

Вычисляет высоту текстовой надписи.

Аргументы:

text	Текстовая строка.
dc	Дескриптор контекста рисования (холста), где планируется разместить надпись. Необязателен.

Возвращает:

Высота надписи.

См. также:

`txTextOut()`, `txSelectFont()`, `txGetTextExtent()`, `txGetTextExtentX()`, `txGetTextExtentY()`

Примеры использования:

```
txTextOut (100 + txGetTextExtentX (text) / 2, 200 + txGetTextExtentY (text) / 2, text);
```

2.8.38. unsigned txSetTextAlign (unsigned align = TA_CENTER|TA_BASELINE, HDC dc = txDC())

Устанавливает текущее выравнивание текста (влево/вправо/по центру).

Аргументы:

align	Флаги выравнивания. Необязательны. Если не указаны, то используются центрирование по горизонтали.
dc	Дескриптор контекста рисования (холста), где планируется разместить надпись. Необязателен.

Возвращает:

Предыдущее состояние выравнивания текста.

Флаги выравнивания:

TA_LEFT	Текст выравнивается влево. Точка (X, Y) определяет левую границу текста.
TA_RIGHT	Текст выравнивается вправо. Точка (X, Y) определяет правую границу текста.
TA_CENTER	Текст центрируется по горизонтали относительно точки (X, Y).
TA_BASELINE	Точка (X, Y) определяет базовую линию текста.
TA_BOTTOM	Точка (X, Y) определяет нижнюю границу текста.
TA_TOP	Точка (X, Y) определяет верхнюю границу текста.

См. также:

`txTextOut()`, `txSelectFont()`, `txGetTextExtent()`, `txGetTextExtentX()`, `txGetTextExtentY()`

Примеры использования:

```
txSetTextAlign     (TA_RIGHT);
txTextOut    (700, 100, "Чтобы доступ был лёгок и быстр, ");
txTextOut    (700, 150, "Переменную клади в регистр.");
txSetTextAlign();
```

2.8.39. LOGFONT* txFontExist (const char name[])

Ищет шрифт по его названию.

Аргументы:

name	Название шрифта.

Возвращает:

Информация о шрифте в структуре `LOGFONT`. Если шрифт не найден, возвращает `NULL`.

См. также:

`txTextOut()`, `txSelectFont()`

Примеры использования:

```
if (txFontExist ("Comic Sans MS"))    txSelectFont ("Comic Sans MS", 30)
else                                  txSelectFont ("Times New Roman", 30);
txTextOut (100, 100, "Комик ли Санс?"); // google.ru/search?q=philosoraptor
```

2.8.40. HDC txCreateCompatibleDC (double sizeX, double sizeY, HBITMAP bitmap = NULL, RGBQUAD **pixels = NULL)

Создаёт дополнительный холст (контекст рисования, Device Context, DC) в памяти.

Аргументы:

sizeX	Ширина холста.
sizeY	Высота холста.
bitmap	Bitmap to be associated with DC (optional). If omit-ted, color bitmap will be created automatically via `Win32::CreateDIBSection`.
pixels	This parameter is optional and may be omitted. When bitmap `== NULL`, a `DIBSection` will be created (see above) and this parameter will be associated with its pixels colors array pointer. See `txCreateDIBSection()` function below. `Also, txCreateDIBSection()` is preferred when using per-pixel transparency.

Возвращает:

Системный номер (по-научному — дескриптор, `handle`), выданный Windows для созданного холста (по-научному — контекста рисования, `Device Context, DC`). Вместе получается `Handle of Device Context == HDC`.

После создания на этом холсте можно рисовать, как и на основном окне **TXLib**, используя параметр `dc` функций рисования. Созданный холст имеет свои собственные настройки цветов, шрифтов и т.д., отличные от настроек основного окна **TXLib**. Чтобы увидеть результат рисования, скопируйте изображение холста на экран с помощью функций `txBitBlt(), txTransparentBlt()` или `txAlphaBlend()`.

Предупреждения:

- *Созданный контекст затем будет нужно обязательно удалить при помощи* `txDeleteDC()`.
- *When the program will be shutting down,* ***TXLib*** *will try to delete DCs which were not deleted, but this is not guaranteed. При этом удаляется и* `bitmap`*, будьте внимательны.*

См. также:

`txCreateWindow()`, `txCreateCompatibleDC()`, `txLoadImage()`, `txDeleteDC()`, `txSaveImage()`, `txGetExtent()`, `txCreateDIBSection()`, `txVideoMemory()`

Примеры использования:

```
HDC save = txCreateCompatibleDC (100, 100);
txBitBlt (save, 0, 0, 100, 100, txDC(), 0, 0); // Сохраняем фон
txTextOut (20, 20, "Boo!");
txSleep (2000);
xBitBlt (txDC(), 0, 0, 100, 100, save, 0, 0); // Текст исчезает
txDeleteDC (save);
```

2.8.41. HDC txCreateDIBSection (doublesizeX, doublesizeY, RGBQUAD **pixels = NULL)

Создаёт аппаратно-независимый дополнительный холст (контекст рисования, `Device Context, DC`) в памяти с возможностью прямого доступа к нему как к массиву.

Аргументы:

sizeX	Ширина холста.
sizeY	Высота холста.
pixels	Указатель на переменную, которая будет использоваться для доступа к пикселям изображения. Необязателен. Эта переменная должна быть указателем на массив структур `RGBQUAD`, каждая из которых описывает цвет одного пикселя. Не надо создавать самому этот массив! Его создаст `txCreateDIBSection()` и вернёт через этот указательный параметр. Объявите только указатель и занулите его, потом его адрес передайте в функцию.

Возвращает:

- Дескриптор созданного аппаратно-независимого холста (контекста рисования).
- После создания на этом холсте можно рисовать, как и на основном окне **TXLib**, используя параметр `dc` функций рисования. Созданный холст имеет свои собственные настройки цветов, шрифтов и т.д., отличные от настроек основного окна **TXLib**. Чтобы увидеть результат рисования, скопируйте изображение холста на экран с помощью функций `txBitBlt()`, `txTransparentBlt()` или `txAlphaBlend()`.
- Самое важное, что аппаратно-независимые холсты, создаваемые этой функцией, позволяют напрямую и быстро изменять цвета пикселей изображения, а также его прозрачность в каждой точке. *См. пример использования.*
- Для прямого доступа к пикселям холста, как к массиву, надо объявить указатель на массив структур `RGBQUAD` и передать адрес этого указателя в качестве третьего параметра функции `txCreateDIBSection()`. Она изменит значение этого указателя так, что он станет указывать на массив цветов пикселей холста.

- Массив `pixels` одномерный, но по сути он описывает двумерное изображение. Поэтому с ним надо работать как с двумерным прямоугольным массивом, физически расположенным в одномерном массиве: вручную вычислять смещение от начала массива до нужного пикселя и после этого адресоваться к массиву. (Так обычно делают, размещая двумерные массивы в динамической памяти.) Кроме того, "Y-ось" этого массива направлена вверх, а не вниз, как в окне **TXLib**. Поэтому для нужного пикселя его смещение от начала массива нужно рассчитывать с помощью формулы `x + (-y + sizeY) * sizeX`. *См. пример использования.*

Предупреждения:

- *Будьте осторожны, не выходите за границы массива, последствия будут непредсказуемыми.*
- *Созданный контекст затем будет нужно обязательно удалить при помощи* `txDeleteDC()`.
- *When the program will be shutting down,* ***TXLib*** *will try to delete* `DCs` *which were not deleted, but this is not guaranteed.*

Заметки:

- Память под массив `RGBQUAD` выделять не надо и освобождать его не надо, этим занимается сама `txCreateDIBSection()` вместе с `txDeleteDC()`.
- Аппаратно-независимые холсты — это контексты устройств, связанные с аппаратно-независимыми растрами (Device Independent Bitmaps, DIB) Windows.

См. также:

`txCreateWindow()`, `txCreateCompatibleDC()`, `txLoadImage()`, `txDeleteDC()`, `txSaveImage()`, `txGetExtent()`, `txCreateCompatibleDC()`

Примеры использования:

```
int main()
    {
    txCreateWindow (800, 600);
    POINT size = txGetExtent();
    txSetFillColor (TX_BLACK);
    txTextCursor (false);
    txBegin();
    HDC src = GetDC (HWND_DESKTOP);                 // Get HDC from Windows Desktop
    RGBQUAD* buf = NULL;                            // Do NOT actually create the array!
    HDC dc = txCreateDIBSection (size.x, size.y, &buf);
    assert (dc); assert (buf);                      // Here, 'buf' points to an array created
                                                    // by txCreateDIBSection()
    while (!GetAsyncKeyState (VK_ESCAPE))
       {
       txBitBlt (dc, 0, 0, size.x, size.y, src);
       for (int y = 0; y < size.y; y++)
       for (int x = 0; x < size.x; x++)
```

```
      {
      RGBQUAD* c = & buf [x + y * size.x];        // Get color at (x, y) within image buffer
      c->rgbRed = (BYTE) c->rgbRed;
      c->rgbGreen = (BYTE) c->rgbGreen;
      c->rgbBlue = (BYTE) c->rgbBlue;
      double r = hypot (x - size.x/2, y - size.y/2);
      double alpha  = sin (0.05 * r) * 0.1 + 0.5;
      c->rgbReserved = (BYTE) ROUND (255 * alpha);       // Set alpha-channel (transparency)
      }
   txUseAlpha (dc) asserted;                            // Premultiply colors with alpha
   txClear();
   txAlphaBlend (txDC(), 0, 0, 0, 0, dc);
   printf ("FPS %.01f\t\t\r", txGetFPS());
   txSleep (0);
   }
txSaveImage ("TxFilter.bmp", dc);

txDeleteDC (dc);
ReleaseDC (HWND_DESKTOP, src);                          // Free Windows Desktop HDC

return 0;
}
```

2.8.42. HDC txLoadImage (const char filename[], int sizeX = 0, int sizeY = 0, unsigned imageFlags = IMAGE_BITMAP, unsigned loadFlags = LR_LOADFROMFILE)

Загружает из файла изображение в формате BMP. Делает это довольно медленно.

Аргументы:

filename	Имя файла с изображением в формате BMP.
sizeX	Размер загружаемого изображения. Необязателен. Если не указан, используется оригинальный размер рисунка из файла.
sizeY	Размер загружаемого изображения. Необязателен. Если не указан, используется оригинальный размер рисунка из файла.
imageFlags	Тип загружаемого изображения, *см. ниже*. Необязателен. Если не указаны, загружается рисунок в формате BMP.
loadFlags	Флаги загрузки изображения, см. ниже. Необязательны. Если не указаны, загрузка идет из файла.

Возвращает:

Дескриптор созданного контекста рисования в памяти, с загруженным изображением. Если изображение не загружено (не найден файл, неверный формат файла и т.д.), то `NULL`.

Предупреждения:

- *Изображение загружается в автоматически создаваемый контекст рисования в памяти ("виртуальный холст"), который затем будет нужно обязательно удалить при помощи* `txDeleteDC()`.

- *When the program will be shutting down,* ***TXLib*** *will try to delete DCs which were not deleted, but this is not guaranteed.*

Заметки:

- Изображения поддерживаются только в формате BMP. Если взять файл другого формата, например JPG, и переименовать его со сменой расширения на BMP, то от этого формат не изменится. Такое изображение загружено не будет.
- Если функция вернула `NULL`, то надо прежде всего проверить наличие файла изображения по указанному в программе пути и формат файла. Если путь к файлу не указан (или указан как неполный), то путь отсчитывается от текущей папки программы, которая может не совпадать текущей папкой среды программирования. Текущую папку программы можно посмотреть по команде `About` в системном меню (она указана там как `"Run from"`).
- Если изображение в файле содержит альфа-канал (информацию о прозрачности), то его цвета должны находиться в формате `Premultiplied Alpha`. *См. замечания* к функции `txAlphaBlend()`.
- Если изображение в файле с альфа-каналом не находится в формате Premultiplied Alpha, то после вызова `txLoadImage()` нужно вызвать функцию `txUseAlpha()`. Однако, не надо этого делать, если цвета в файле уже находятся в формате Premultiplied Alpha, иначе картинка станет темнее. Также не надо вызывать `txUseAlpha()` несколько раз для одного и того же изображения.
- Не надо часто загружать одно и то же изображение, особенно в цикле. От этого программа начинает тормозить!

Загрузите один раз перед циклом, потом используйте много раз. *Посмотрите, как это сделано в примере* `TX\Examples\Tennis\Tennis.cpp`.

Типы загружаемых изображений:

IMAGE_BITMAP	Рисунок в формате BMP
IMAGE_CURSOR	Курсор в формате CUR или ANI
IMAGE_ICON	Иконка в формате ICO

Флаги загрузки:

LR_ CREATEDIBSECTION	Создаёт DIB (device-independent bitmap), удобную для прямого доступа к данным
LR_LOADFROMFILE	Загружает из файла (если этот флаг не указан, то загружает из ресурса **EXE**-файла)
Остальные флаги загрузки	*см. на* MSDN.com, поиск "LoadImage function"

См. также:

`txCreateWindow()`, `txCreateCompatibleDC()`, `txLoadImage()`, `txDeleteDC()`, `txBitBlt()`, `txAlphaBlend()`, `txTransparentBlt()`, `txSaveImage()`, `txGetExtent()`

Примеры использования:

```
HDC background_CopiedFromHelp = txLoadImage ("Resources\\Images\\Background.bmp");
if (!background_CopiedFromHelp)
    txMessageBox ("Не могу загрузить фон из Background.bmp", "Да, я скопировал это из примера");
// Не надо часто загружать одно и то же изображение, особенно в цикле — программа будет тормозить!
// Загрузите один раз перед циклом, потом используйте много раз.
// Посмотрите, как сделано в примере TX\Examples\Tennis\Tennis.cpp.
txBitBlt (txDC(), 0, 0, 800, 600, background_CopiedFromHelp, 0, 0);
...
txDeleteDC (background_CopiedFromHelp);
```

2.8.43. bool txDeleteDC (HDCdc)

Уничтожает холст (контекст рисования, DC) в памяти.

Аргументы:

dc	Контекст рисования для уничтожения.*Если передан указатель на контекст, то после уничтожения по указателю записывается* NULL.

Возвращает:

Если операция была успешна — true, иначе — false.

Заметки:

Эту функцию нужно обязательно вызывать, если картинка, загруженная с помощью txLoadImage() или виртуальный холст, созданный с помощью txTransparentBlt() или

txCreateDIBSection(), больше не нужны. Иначе ~~настанет полный трешь~~ у **Windows** могут закончиться свободные дескрипторы HDC, и начнётся общее торможение всей работы **Windows** и проблемы с рисованием во всех программах. Может быть, поосле этого придётся перезапускать компьютер.

См. также:

txCreateWindow(), txCreateCompatibleDC(), txLoadImage(), txDeleteDC(), txSaveImage(), txGetExtent(), txCreateDIBSection()

Примеры использования:

```
HDC background_CopiedFromHelp = txLoadImage ("Resources\\Images\\Background.bmp");

if (!background_CopiedFromHelp)
   txMessageBox ("Не могу загрузить фон из Background.bmp, и я скопировал это из примера");

// См. важный комментарий в примере к функции txLoadImage!
txBitBlt (txDC(), 0, 0, 800, 600, background_CopiedFromHelp, 0, 0);

...

txDeleteDC (background_CopiedFromHelp);
```

2.8.44. bool txBitBlt (HDC destImage, double xDest, double yDest, double width, double height, HDC sourceImage, double xSource = 0, double ySource = 0, unsigned operation = SRCCOPY)

Копирует изображение с одного холста (контекста рисования, DC) на другой.

Аргументы:

destImage	Контекст назначения (куда копировать). Для копирования в окно **TXLib** укажите `txDC()`.
xDest	X-координата верхнего левого угла копируемого изображения.
yDest	Y-координата верхнего левого угла копируемого изображения.
width	Ширина копируемой области. Если равна `0`, то равна ширине изображения-источника.
height	Высота копируемой области. Если равна `0`, то равна высоте изображения-источника.
sourceImage	Контекст источника (откуда копировать).
xSource	X-координата верхнего левого угла копируемой области внутри изображения-источника. Необязательна. Если не указана, то `0`.
ySource	Y-координата верхнего левого угла копируемой области внутри изображения-источника. Необязательна. Если не указана, то `0`.
operation	Вид операции копирования. Список видов гуглите по запросу "`BitBlt function MSDN`". Необязательен. Если не указан, то `SRCCOPY` (обычное копирование изображения).

Возвращает:

Если операция была успешна — `true`, иначе — `false`.

Предупреждения:

Если контексты источника равен `NULL`, то он не существует и копирование вызовет ошибку. Наиболее частая причина — ошибка при загрузке файла изображения и отсут-

ствие проверки на эту ошибку. Пример с проверкой на правильность загрузки см. ниже.

См. также:

txAlphaBlend(), txTransparentBlt(), txSaveImage(), txGetExtent(), txSetColor(), txGetColor(), txSetFillColor(), txGetFillColor(), txColors, RGB(), txCreateDIBSection()

Примеры использования:

Пример использования см. в файле TX\Examples\Tennis\Tennis.cpp.

```
HDC background_CopiedFromHelp = txLoadImage ("Resources\\Images\\Background.bmp");
   if (!background_CopiedFromHelp)
("Не могу фон из загрузить Background.bmp, и да, я взял этот код из примера");
// См. важный комментарий в примере к функции txLoadImage!
txBitBlt (txDC(), 0, 0, 800, 600, background_CopiedFromHelp);
...
txDeleteDC (background_CopiedFromHelp);
```

2.8.45. bool txBitBlt (double xDest, double yDest, HDC sourceImage, double xSource = 0, double ySource = 0)

Копирует изображение на экран.

Аргументы:

xDest	X-координата верхнего левого угла копируемого изображения.
yDest	Y-координата верхнего левого угла копируемого изображения.

sourceImage	Копируемое изображение.
xSource	X-координата верхнего левого угла копируемой области внутри изображения-источника. Необязательна. Если не указана, то `0`.
ySource	Y-координата верхнего левого угла копируемой области внутри изображения-источника. Необязательна. Если не указана, то `0`.

Возвращает:

Если операция была успешна – `true`, иначе – `false`.

См. описание в функции `txBitBlt()` *выше.*

2.8.46. `bool txTransparentBlt (HDC destImage, double xDest, double yDest, double width, double height, HDC sourceImage, double xSource = 0, double ySource = 0, COLORREF transColor = TX_BLACK)`

Копирует изображение с одного холста (контекста рисования, `DC`) на другой с учётом прозрачности.

Аргументы:

destImage	Контекст назначения (куда копировать). Для копирования в окно **TXLib** укажите `txDC()`.
xDest	X-координата верхнего левого угла копируемого изображения.
yDest	Y-координата верхнего левого угла копируемого изображения.
width	Ширина копируемой области. Если равна `0`, то автоматически берётся из размера источника.
height	Высота копируемой области. Если равна `0`, то автоматически берётся из размера источника.

sourceImage	Контекст источника (откуда копировать).
xSource	X-координата верхнего левого угла копируемой области внутри изображения-источника. Необязательна. Если не указана, то `0`.
ySource	Y-координата верхнего левого угла копируемой области внутри изображения-источника. Необязательна. Если не указана, то `0`.
transColor	Цвет, который будет считаться прозрачным. Необязателен. Если не указан, то `TX_BLACK`.

Возвращает:

Если операция была успешна — `true`, иначе — `false`.

Предупреждения:

- *Если контекст источника равен* `NULL`*, то он не существует и копирование вызовет ошибку. Наиболее частая причина – ошибка при загрузке файла изображения и отсутствие проверки на эту ошибку. Пример с проверкой на правильность загрузки см. ниже.*
- *Изображение-источник и изображение-приемник не могут налагаться друг на друга.*
- *Если прямоугольник копируемой области не полностью лежит внутри изображения-источника, то функция работать не будет. Такое бывает, если* `xSource` *или* `ySource` *отрицательны, или величина* `(xSource + width)` *больше ширины изображения-источника, или величина* `(ySource + height)` *больше высоты изображения-источника.*
- *Стандартная функция* `TransparentBlt` *из* `Win32 API` *может масштабировать изображение. В* `txTransparentBlt` *это убрано для упрощения использования. If you need image scaling, use original function* `TransparentBlt` *and don't mess with stupid*

TX-based tools. (See implementation of `txTransparentBlt` *in* `TXLib.h`*).*

Заметки:

- Если `TransparentBlt` не работает, используйте функцию `AlphaBlend`, она вообще лучше.
- Стандартная функция `Win32::TransparentBlt` из `Win32 API` может масштабировать изображение. В `txTransparentBlt` это убрано для упрощения использования. If you still need image scaling, use original function **Win32**::TransparentBlt and don't mess with stupid TX-based tools. (See implementation of `txTransparentBlt` in **TX library** source: open `TXLib.h` file in your editor and search for `txTransparentBlt` function definition.)

См. также:

`txBitBlt()`, `txAlphaBlend()`, `txLoadImage()`, `txCreateCompatibleDC()`, `txSaveImage()`, `txGetExtent()`, `txCreateDIBSection()`

Примеры использования:

Пример использования *см. в файле* `TX\Examples\Tennis\Tennis.cpp`.

```
HDC superman_CopiedFromHelp = txLoadImage ("Resources\\Images\\Superman.bmp");

if (!superman_CopiedFromHelp)

   txMessageBox ("Cannot load superman, all the monsters will succeed");

// См. важный комментарий в примере к функции txLoadImage!

txTransparentBlt (txDC(), 0, 0, 800, 600, superman_CopiedFromHelp);

// А можно и так:

Win32::TransparentBlt (txDC(), 0, 0, 800, 600, superman, 0, 0, 80, 60, -1)

// Познай мощь Win32 GDI, отказавшись от TXLib'a! :) см. TransparentBlt в MSDN.com.

...
```

```
txDeleteDC (superman_CopiedFromHelp); // So pity :( But he was only a copy
```

2.8.47. bool txTransparentBlt (double xDest, double yDest, HDC sourceImage, COLORREF transColor = TX_BLACK, double xSource = 0, double ySource = 0)

Копирует изображение на экран с учетом прозрачности.

Аргументы:

xDest	X-координата верхнего левого угла копируемого изображения.
yDest	Y-координата верхнего левого угла копируемого изображения.
sourceImage	Копируемое изображение.
transColor	Цвет, который будет считаться прозрачным. Необязателен. Если не указан, то TX_BLACK.
xSource	X-координата верхнего левого угла копируемой области внутри изображения-источника. Необязательна. Если не указана, то `0`.
ySource	Y-координата верхнего левого угла копируемой области внутри изображения-источника. Необязательна. Если не указана, то `0`.

Возвращает:

Если операция была успешна — `true`, иначе — `false`.

См. описание в функции `txTransparentBlt()` выше.