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 3

Фонд развития Физтех-лицея и Физтех-школ

Иннопрактика

МФТИ
Долгопрудный, 2021

СОДЕРЖАНИЕ

5. ДИАЛОГОВЫЕ ОКНА

5.1. Классы

- `struct txDialog`

  Базовый класс для диалоговых окон.

- `struct txDialog::Layout`

  Класс для описания элемента диалогового окна (контрола)

5.2. Перечисления

- `enum CONTROL { DIALOG = (int) 0x00000000, BUTTON = (int) 0xFFFF0080, EDIT = (int) 0xFFFF0081, STATIC = (int) 0xFFFF0082, LISTBOX = (int) 0xFFFF0083, SCROLLBAR = (int) 0xFFFF0084, COMBOBOX = (int) 0xFFFF0085, END = (int) 0x00000000 }`

  Константы для задания типа контрола.

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.

5.3. Функции

5.3.1. const char* txInputBox (const char *text = NULL, const char *caption = NULL, const char *input = NULL)

Ввод строки в отдельном окне.

Аргументы:

text	Текст с вопросом. Необязательно.
caption	Заголовок окна. Необязательно.
input	Значение строки по умолчанию. Необязательно.

Возвращает:

Введённая строка (статическая переменная функции).

Предупреждения:

Возвращаемая строка – статическая, и обновляется при каждом вызове функции. Если `txInputBox()` будет вызываться несколько раз, то для сохранения строки её необходимо копировать в другую строку при помощи strcpy().

См. также:

`txDialog, TX_BEGIN_MESSAGE_MAP, TX_BEGIN_COMMAND_MAP, TX_HANDLE, TX_END_MESSAGE_MAP`

Примеры использования:

```
const char* name = txInputBox ("So what's ur name?!?!", "System);
txMessageBox (name, "Aaand noooowww.. the winner iiis...");
```

5.4. Перечисления

5.4.1. enum CONTROL [inherited]

Константы для задания типа контрола.

Вместо констант можно использовать названия оконных классов, преобразованные к типу `txDialog::CONTROL`.

См. также:

`txDialog::Layout, txDialog::setLayout()`

Примеры использования:

См. реализацию функции `txInputBox()`.

Элементы перечислений:

DIALOG	Начало описания диалога
BUTTON	Кнопка
EDIT	Редактируемый текст
STATIC	Нередактируемый элемент (текст, картинка и т.д.)
LISTBOX	Список с прокруткой
SCROLLBAR	Полоса прокрутки
COMBOBOX	Комбинированный список
END	Конец описания диалога

6. ТЕХНИЧЕСКИЕ ДЕТАЛИ

6.1. 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

6.2. Инициализация библиотеки

- `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**.

6.3. Настроечные константы и переменные

- `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.

- `#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_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.

- `#define _TX_FATAL_EXCEPTIONS_LIMIT 16`

  Максимальное количество фатальных исключений.

6.4. Внутренняя диагностика

- `#define _TX_ALLOW_TRACE`

  Включает/отключает внутреннюю трассировку исполнения кода библиотеки.

- `#define TX_TRACE`

  Трассирует исполнение кода через OutputDebugString().

6.5. Макросы

- `#define _TX_VER`

  Текущая версия библиотеки.

Версия библиотеки в целочисленном формате: старшее слово — номер версии, младшее — номер ревизии, в двоично-десятичном формате. Например, `0x172a0050` — версия `0.172a`, ревизия `50`.

```
#define _TX_VERSION "TXLib [Ver: 1.73a, Rev: 164, Date: 2020-01-30 05:00:00 +0300]"
#define _TX_AUTHOR "Copyright (C) Ded (Ilya Dedinsky, http://txlib.ru)" // ПРИМЕР
#define _TX_VER 0x173a0000
```

Эти константы автоматически обновляются при изменении версии.

См. также:

`txVersion()`

Примеры использования:

```
#if !(defined (_TX_VER) && (_TX_VER >= 0x172a0000))
#error Must use TXLib.h version >= 1.72 to compile this.
#endif
```

6.5.1. #define _TX_MODULE

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

Заметки:

Можно переопределять до включения файла `TXLib.h`.

6.5.2. #define _TX_NOINIT

Запрет ранней инициализации **TXLib**.

Если константа определена с помощью `#define` до включения `TXLib.h` в программу, ранняя инициализация (до запуска функции `main`) не проводится.

Заметки:

Ранняя инициализация включает:

- Открытие окна консоли,
- Установку кодовой страницы `_TX_CODEPAGE (1251)` консоли для поддержки русского языка,
- Установку русской локали стандартной библиотеки **C++** (`"Russian" / "ru_RU.CP1251"` и `L"Russian_Russia.ACP"`),
- Переинициализация библиотек `stdio` и `iostream` на случай, если приложение не скомпоновано как консольное и библиотеки остались неинициализированы,
- Перехват системных сигналов (деление на `0`, арифметические ошибки, обращение по неверному адресу и т.д.),
- Перехват необработанных исключений,
- Смена заголовка консольного окна,
- Установка режима паузы по завершении программы, чтобы окно закрывалось не сразу,
- Подавление паузы при запуске из сред программирования, заставляющей нажимать на клавишу два раза.
- Если ранняя инициализация запрещена, но в `EXE`-файле создается окно **TXLib** с помощью `txCreateWindow()`, то библиотека всё равно будет инициализирована. В `DLL` ранняя инициализация никогда не проводится.
- Ранняя инициализация не потокобезопасна (`not thread-safe`).

См. также:

`txCreateWindow()`, `_TX_ALLOW_KILL_PARENT`, `_TX_WAITABLE_PARENTS`, `_txWatchdogTimeout`, `_txConsoleMode`

Примеры использования:

```
#define _TX_NOINIT
#include "TXLib.h"
```

6.5.3. #define TX_USE_SFML

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

Заметки:

Этот макрос задаётся перед включением `TXLib.h` в программу.

Примеры использования:

```
#define TX_USE_SFML
#include "TXLib.h"
#include <SFML/Graphics.hpp>
#define _TX_WAITABLE_PARENTS
"Winpty-agent.exe:Clion.exe,          " /* 0: CLion32 */ \
"Winpty-agent.exe:Clion64.exe,        " /* 1: CLion64 */ \
"starter.exe:eclipse.exe,             " /* 2: Eclipse 4 */ \
"starter.exe:javaw.exe,               " /* 3: Eclipse 3 */ \
"cmd.exe:devenv.exe,                  " /* 4: MSVS 2003+ */ \
"VSDebugConsole.exe:devenv.exe,       " /* 5: MSVS 2019 */ \
"consolepauser.exe:devcpp.exe,        " /* 6: Dev-Cpp */ \
```

6.5.4. "cb_console_runner.exe:codeblocks.exe"

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

Если программа перечислена в списке и **TXLib** запущена из неё, то при завершении **TXLib** указанная программа будет закрыта. (Это произойдёт, если не открыто графическое окно **TXLib**, а есть только окно консоли.)

Программы разделяются пробелом или запятой. Допускается указание родителя запускающей программы, после двоеточия. Имя программы, начинающееся с большой буквы — спец. случай для консоли **CLion**, *см. функции* `_txCleanup()` и `_txIsParentWaitable()` в исходном тексте `TXLib.h`.

Может задаваться перед включением `TXLib.h` в программу.

См. также:

`_TX_ALLOW_KILL_PARENT`, `_TX_NOINIT`, `_txCleanup()`, `_txIsParentWaitable()`, `_txWatchdogTimeout`

6.5.5. #define _TX_ALLOW_KILL_PARENT true

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

Иначе отменяется собственная пауза до нажатия клавиши, встроенная в **TXLib**, и пусть тогда паузу делает вызывающий процесс.

Список вызывающих программ, которые могут делать такую паузу, задаётся в `_TX_WAITABLE_PARENTS`.

Заметки:

Макрос должен задаваться перед включением `TXLib.h` в программу.

См. также:

`_TX_WAITABLE_PARENTS, _TX_NOINIT, _txWatchdogTimeout`

Примеры использования:

```
#define _TX_ALLOW_KILL_PARENT false
#include "TXLib.h"
```

6.5.6. #define TX_COMPILED

Макросы для поддержки прекомпиляции **TX Library.**

Не секрет, что файл `TXLib.h` очень большой и, как следствие, долго компилируется. Для ускорения сборки проектов обычно используется техника раздельной компиляции, когда в проект входят не один, а несколько `.cpp`-файлов. В этом случае, если изменён только один такой файл, перекомпилироваться будет только он, а результаты компиляции остальных файлов присоединяются на этапе линковки, что гораздо быстрее.

Однако библиотека **TX Library** представляет собой единственный заголовочный `.h`-файл, а заголовочные файлы перекомпилируются каждый раз, при каждой сборке. Это существенно замедляло работу с библиотекой.

Теперь файл `TXLib.h` внутри разделен на части, которые могут по-разному вести себя при компиляции.

Если при компиляции перед включением файла `TXLib.h` определён макрос `TX_COMPILED`, то компилироваться будет меньшая часть файла, порядка одной трети. Эта часть содержит то, что обязательно надо включать в каждую компиляцию: определения констант, структур и классов, прототипы функций, шаблоны, макросы и др. При этом около двух третей файла, которые содержат определения нешаблонных функций,

не будут компилироваться. (Строго говоря, код этих двух третей обычно не помещают в заголовочные файлы, но, так как библиотека построена по принципу единственного файла для упрощения работы с ней, такой код она вынуждена содержать.)

Чтобы в проекте были доступны откомпилированные определения всех необходимых функций библиотеки, в него нужно добавить ещё один файл, включив в него `TXLib.h` и определив в нём макрос `TX_COMPILING`. Это приведёт к компиляции всего содержимого библиотеки, и её машинный код станет доступен всей программе. В примере ниже этот файл назван `TXLib.cpp`, но имя может быть любое. Для компиляции этого файла потребуется время, но затем он перестанет участвовать в компиляции, так как его содержимое не будет изменяться, и общее время сборки проекта сократится, от 1.5 до 3 раз.

Предупреждения:

Не следует помещать в этот файл ничего, кроме директив `#define TX_COMPILING` и `#include "TXLib.h"`, так как при изменении содержимого этого файла среда программирования снова перекомпилирует его, на что вновь потребуется большое время.

Также для ускорения компиляции можно определить макрос `WIN32_LEAN_AND_MEAN` до включения `TXLib.h` в программу.

Предупреждения:

Макрос `WIN32_LEAN_AND_MEAN` *задаётся автоматически при задании макроса* `TX_COMPILED`. *Если при этом не видны определения каких-либо сущностей, определённых в* `Windows.h` *(функций, структур, классов, констант и т.п.), включите в программу файлы, определяющие эти сущности, вручную.*

Заметки:

Если определены макросы TX_COMPILED или TX_COMPILING, анонимное пространство имён, окружающее пространство имён TX, не используется.

Примеры использования:

```
// Оба файла ниже должны входить в один и тот же проект Visual Studio или другой среды
// программирования (IDE). Для добавления файла в проект используйте возможности вашей среды
// программирования.

//-----------------------------------------------
// Main.cpp: Файл с текстом программы
//-----------------------------------------------

#define TX_COMPILED // Обязательно ПЕРЕД командой #include
#include "TXLib.h"

int main()
      {
      txCreateWindow (800, 600);
    ...
      }

//-----------------------------------------------
// TXLib.cpp: Файл для прекомпиляции TXLib
//-----------------------------------------------

#define TX_COMPILING // Обязательно ПЕРЕД командой #include
#include "TXLib.h"

// [Больше в этом файле ничего нет]
```

6.5.7. #define _TX_ALLOW_TRACE

Включает/отключает внутреннюю трассировку исполнения кода библиотеки.

Трассировка идет через макрос `TX_TRACE`, который подставляется перед каждым оператором (`statement`). По умолчанию трассировка выключена.

По умолчанию трассировка ведётся через функцию `OutputDebugString()`, её вывод можно перехватить утилитами-логгерами, например, `DbgView`. Это можно изменить, переопределив макрос `TX_TRACE` до включения `TXLib.h` в программу.

Предупреждения:

*Трассировка очень тормозит выполнение программы, особенно при отладке в **Microsoft Visual Studio**. В этом случае лучше пользоваться **DbgView** (см. выше) и запускать отлаживаемую программу отдельно, а не из-под отладчика **Visual Studio.***

Заметки:

`_TX_ALLOW_TRACE` и `TX_TRACE` задаются перед включением `TXLib.h` в программу.

Уровни трассировки:

1	Regular functions
2	Reserved
3	Init/Cleanup
4	Init/Cleanup, misc functions
5	Error handling, entry points
6	Error handling, main part

7	Error handling, misc functions
8	Canvas worker thread
9	Reserved

Примеры использования:

```
// Для просмотра трассы запустите DbgView ДО запуска программы!

#define _TX_ALLOW_TRACE 1 // Трассировать только обычные функции TXLib (уровень 1).
#include "TXLib.h"

#define _DEBUG // Не трассировать, но собирать информацию о вызовах
#include "TXLib.h" // функций TXLib для возможной трассировки стека.
```

6.5.8. #define TX_TRACE

Трассирует исполнение кода через `OutputDebugString()`.

По умолчанию трассировка ведётся через функцию `OutputDebugString()`, её вывод можно перехватить утилитами-логгерами, например, **DbgView**. Для "раскраски" лога есть файл `TX\Dev\DbgView.ini`, его надо загрузить в **DbgView** через меню `Edit/Filter/Load (Ctrl+L)`.

С помощью `TX_TRACE` можно трассировать код самой библиотеки **TXLib**. Для этого надо разрешить трассировку **TXLib**, определив макрос `_TX_ALLOW_TRACE` перед включением файла `TXLib.h` в программу. По умолчанию трассировка **TXLib** выключена.

`TX_TRACE` можно переопределить в свой код. Тогда, если трассировка библиотеки разрешена, он будет вызываться почти перед каждой исполняемой строкой **TXLib**. Может быть, это кому-нибудь будет интересно.

Примеры использования:

```
int main()
      {
    ...
      TX_TRACE; // Через DbgView увидим имя файла и номер выполняемой строки
    ...
      }
#define TX_TRACE printf (__TX_FILELINE__ "\t" __TX_FUNCTION__ "\n");
#include "TXLib.h"
```

6.6. Функции

6.6.1. const char* txVersion ()

Возвращает строку с информацией о текущей версии библиотеки.

Возвращает:

Строка с информацией о текущей версии библиотеки.

Примеры использования:

```
printf ("I personally love %s\n", txVersion());
```

6.6.2. unsigned txV ersionNumber ()

Возвращает номер версии библиотеки.

Возвращает:

Номер версии библиотеки.

Примеры использования:

```
printf ("My magic number is %x\n", txVersionNumber());
```

6.6.3. const char* txGetModuleFileName (bool fileNameOnly = true)

Возвращает имя исполняемого файла или изначальный заголовок окна TXLib.

Аргументы:

fileNameOnly	Возвратить только полное имя исполняемого файла, полученного через **Win32** функцию `GetFileModuleName (NULL, ...)`. Необязательно. Если не указано, то возвращается полное имя исполняемого файла.

Возвращает:

`fileNameOnly = true:` Имя исполняемого файла.

`fileNameOnly = false:` Изначальный заголовок окна TXLib.

Заметки:

Возвращается статическая строка.

См. также:

`txWindow(), txVersion(), txVersionNumber()`

Примеры использования:

```
printf ("Смотрите на заголовок окна!");

  for (int done = 0; done <= 100; done++)
      {
      char title [1024] = "";
```

```
        sprintf (title, "%s - [%-10.*s] %d%%", txGetModuleFileName (false), done/10);

        SetWindowText (txWindow(), title);
        SetWindowText (Win32::GetConsoleWindow(), title);
        txSleep (50);
        }

    txMessageBox ("Вот такой вот progress bar", "TXLib forever)");
```

6.7. Переменные

6.7.1. char _txLogName = ""

Имя лог-файла **TXLib**.

В эту строку надо скопировать нужное вам имя лог-файла.

По умолчанию файл создаётся во время ошибки в одной папке с запущенной программой, имеет то же имя, что и файл программы, с добавлением расширения ".log".

Заметки:

Устанавливать имя лог-файла надо в начале работы программы, до появления первой ошибки.

6.7.2. int _txConsoleMode = SW_HIDE

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

По умолчанию: SW_HIDE – Скрывать консольное окно.

Заметки:

Переменная устанавливается до открытия окна библиотеки.

См. также:

`_TX_NOINIT`

Примеры использования:

```
_txConsoleMode = SW_HIDE; // Всегда скрывать консольное окно
txCreateWindow (800, 600);

_txConsoleMode = SW_SHOW; // Всегда показывать консольное окно
txCreateWindow (800, 600);
```

6.7.3. int _txWindowStyle = WS_POPUP | WS_BORDER | WS_CAPTION | WS_SYSMENU

Стиль графического окна библиотеки.

Заметки:

Переменная устанавливается до открытия окна библиотеки.

Примеры использования:

```
_txWindowStyle &= ~WS_CAPTION; // FullScreen: окно без заголовка, размером с экран
txCreateWindow (GetSystemMetrics (SM_CXSCREEN), GetSystemMetrics (SM_CYSCREEN));

printf ("Закройте меня через Alt+F4\n");
```

6.7.4. const char * _txConsoleFont = "Lucida Console"

Шрифт консоли

Заметки:

Переменная устанавливается до открытия окна библиотеки.

6.7.5. unsigned _txWindowUpdateInterval = 25

Интервал обновления холста (мс)

Заметки:

Переменная устанавливается до открытия окна библиотеки.

6.7.6. 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`.

Если равен `NULL`, то используется `Win32::BitBlt`.

Вы можете переопределить значение этого указателя, чтобы он указывал на вашу функцию, и тогда **TXLib** будет выводить изображение в окно через нее. Имейте в виду, что эта функция будет вызываться в отдельном потоке.

Параметры этой функции гуглите: "`StretchBlt function`".

См. также:

`txSetWindowsHook()`, `txBitBlt()`, `txAlphaBlend()`

Примеры использования:

```
#include "TXLib.h"

bool MySwapBuffers      (HDC dest, int xDest, int yDest, int wDest, int hDest,
                         HDC src, int xSrc, int ySrc, int wSrc, int hSrc, DWORD rOp)
            {
            return txAlphaBlend (dest, xDest, yDest, wSrc-xSrc, hSrc-ySrc, src, xSrc, ySrc, 0.01);
            }

      int main()
            {
            _txSwapBuffers = MySwapBuffers; // That's it

      txCreateWindow (800, 600);

      txSetFillColor (TX_NULL);
      txRectangle (10, 10, txGetExtentX() - 10, txGetExtentY() - 10);

      for (int x = 50; x <= txGetExtentX() - 50; x += 20)
            {
            txCircle (x, txGetExtentY()/2, 10);
            txSleep (100);
            }

      txSleep (3000);
      }
```

6.7.7. int _txWatchdogTimeout = 10*_TX_TIMEOUT

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

Если значение меньше 0, то время не лимитируется.

Заметки:

Для предотвращения зависания программы при выходе ***TXLib*** запускает отдельный сторожевой поток (watchdog thread), который ожидает _txWatchdogTimeout миллисекунд, а затем принудительно завершает программу.

См. также:

_TX_WAITABLE_PARENTS, _TX_NOINIT

7. КЛАССЫ

7.1. Структура txDialog::Layout

Класс для описания элемента диалогового окна (контрола)

7.1.1. Открытые атрибуты

- CONTROL wndclass

  Тип контрола

- const char * caption

  Название или текст

- `WORD id`
  Идентификатор контрола
- `short x`
  Координата верхнего левого угла
- `short y`
  Координата нижнего правого угла
- `short sx`
  Размер по X.
- `short sy`
  Размер по Y.
- `DWORD style`
  Стиль контрола
- `const char * font`
  Шрифт диалогового окна
- `WORD fontsize`
  Размер шрифта диалогового окна

7.1.2. Подробное описание

Класс для описания элемента диалогового окна (контрола)

Массив таких структур задаёт описание макета диалогового окна. Это описание похоже на задание диалога в ресурсном скрипте (`.rc`):

- Нулевой элемент описывает диалоговое окно в целом

- Остальные элементы описывают контролы (порядок задания параметров контрола похож на порядок параметров в ресурсном скрипте)
- Последний элемент — `txDialog::END` или `{NULL}`

См. также:

`txDialog::setLayout()`, `txDialog::dialogBox()`, `txDialog::dialogProc()`

Примеры использования:

См. реализацию функции `txInputBox()`.

7.2. Класс `txAutoLock`

Класс для автоматической блокировки и разблокировки критической секции.

7.2.1. Открытые члены

- `txAutoLock (CRITICAL_SECTION *cs, bool mandatory=true)`

  Конструктор, блокирует критическую секцию
- `txAutoLock (bool mandatory=true)`

  Конструктор для блокировки холста **TXLib**.
- `~txAutoLock ()`

  Деструктор, разблокирует секцию
- `operator bool () const`

  Позволяет проверить, заблокировалась секция или нет

7.2.2. Открытые атрибуты

- `CRITICAL_SECTION * cs_`

  Блокируемая критическая секция

7.2.3. Закрытые члены

- `txAutoLock (const this_t &) _tx_delete`

  Такой вот копирайт.

7.2.4. Подробное описание

Класс для автоматической блокировки и разблокировки критической секции.

Начиная с Александреску, его пишут все и он прост, как пробка: в конструкторе — `EnterCriticalSection()`, в деструкторе — `LeaveCriticalSection()`. Это **RAII** в чистом виде: вы никогда не забудете разблочить секцию and your thread show will always go on.

Некоторые добавления:

есть возможность вызвать `TryEnterCriticalSection()`, и, если она не заблочила секцию, деструктор её не разблочивает. Есть оператор для проверки, заблочилась ли секция или нет (*см. конструкторы* класса и оператор `bool`).

Заметки:

Класс не инициализирует и не удаляет критическую секцию. Только синхронизирует. Остальное сами-сами :)

См. также:

`txLock()`, `txUnlock()`, `txGDI()`

7.2.5. Конструктор(ы)

7.2.6. txAutoLock (CRITICAL_SECTION *cs, bool mandatory = true) [explicit]

Конструктор, блокирует критическую секцию

Аргументы:

cs	Критическая секция для блокировки.
mandatory	Если true, то блокировать обязательно (EnterCriticalSection). Если false, то только пытаться блокировать (TryEnterCriticalSection). Если не указано, то блокировка обязательна.

Примеры использования:

```
CRITICAL_SECTION cs = {};              // This is not a Counter Strike

void foo()
      {
      txAutoLock lock (&cs);           // Здесь вызовется EnterCriticalSection()
    ...
      }                                // а здесь LeaveCriticalsection()

    void bar()
      {
      txAutoLock lock (&cs, false);    // TryEnterCriticalSection()
      if (!lock) return;               // ну не смогла
    ...
      }
```

7.2.7. txAutoLock (bool mandatory = true) [explicit]

Конструктор для блокировки холста **TXLib**.

Аргументы:

mandatory	Если true, то блокировать обязательно, как в `txLock (true)`. Если false, то только пытаться блокировать, как в `txLock (false)`. Если не указано, то блокировка обязательна.

Примеры использования:

```
void foobar()
      {
      txAutoLock lock;   // Здесь вызовется txLock()
    ...
      }                  // а здесь txUnlock()
```

7.2.8. Методы

7.2.9. operator bool () const

Позволяет проверить, заблокировалась секция или нет

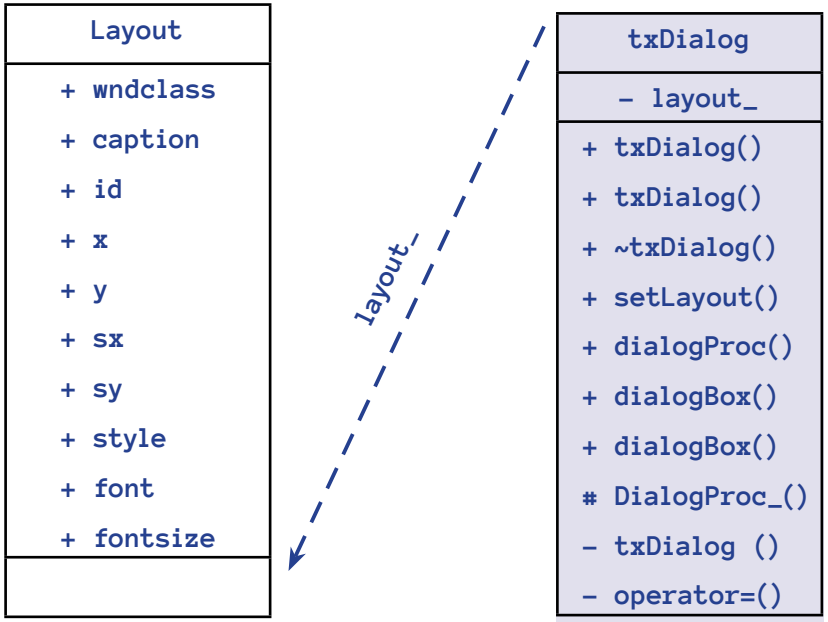
Примеры использования:

См. в `txAutoLock::AutoLock (CRITICAL_SECTION&, bool)`

7.3. Структура txDialog

Базовый класс для диалоговых окон.

Граф связей класса txDialog:



7.3.1. Классы

- struct Layout

Класс для описания элемента диалогового окна (контрола)

7.3.2. Открытые типы

- `enum CONTROL { DIALOG = (int) 0x00000000, BUTTON = (int) 0xFFFF0080, EDIT = (int) 0xFFFF0081, STATIC = (int) 0xFFFF0082, LISTBOX = (int) 0xFFFF0083, SCROLLBAR = (int) 0xFFFF0084, COMBOBOX = (int) 0xFFFF0085, END = (int) 0x00000000 }`

  Константы для задания типа контрола.

7.3.3. Открытые члены

- `txDialog ()`

  Конструктор.
- `txDialog (const Layout *layout)`

  Конструктор.
- `virtual ~txDialog ()`

  Деструктор.
- `const Layout * setLayout (const Layout *layout)`

  Устанавливает текущий макет диалогового окна.
- `virtual int dialogProc (HWND _wnd, UINT _msg, WPARAM _wParam, LPARAM _lParam)`

  Функция обработки сообщений диалогового окна.
- `intptr_t dialogBox (const Layout *layout=NULL, size_t bufsize=0)`

  Запускает диалоговое окно.
- `intptr_t dialogBox (WORD resource)`

  Запускает диалоговое окно.

7.3.4. Защищенные статические члены

- `static intptr_t CALLBACK DialogProc_ (HWND wnd, UINT msg, WPARAM wParam, LPARAM lParam)`

  Настоящая диалоговая функция (не `txDialog::dialogProc()`, т.к. функция окна **Win32** должна быть статической).

7.3.5. Закрытые члены

- `txDialog (const this_t &) _tx_delete`

  Закрытые конструктор копирования и оператор присваивания.

7.3.6. Закрытые данные

- `const Layout * layout_`

  Текущий макет диалога.

7.3.7. Подробное описание

- Базовый класс для диалоговых окон.
- Для создания своего диалогового окна нужно:
- Унаследовать свой класс из этого базового класса;
- Задать состав и расположение элементов управления (контролов) функцией `txDialog::setLayout()`, или указать карту расположения при показе диалогового окна функцией `txDialog::dialogBox()`;
- Переопределить в своем классе функцию `txDialog::dialogProc()` для обработки событий диалогового окна или построить карту реакций на команды с помощью макросов `TX_BEGIN_MESSAGE_MAP()`, `TX_END_MESSAGE_MAP`, `TX_COMMAND_MAP`.

Заметки:

Этот класс предоставляет только базовую функциональность диалоговых окон и только базовые сведения по ним. Для его полноценного использования нужно знать, как работать с диалоговыми окнами в Windows.

См. также:

`txDialog::setLayout()`, `txDialog::dialogProc()`, `txDialog::Layout`, `TX_BEGIN_MESSAGE_MAP()`, `TX_END_MESSAGE_MAP`, `TX_COMMAND_MAP`

Примеры использования:

См. реализацию функции `txInputBox()`.

7.3.8. Конструкторы

7.3.9. txDialog ()

Конструктор.

См. также:

`txDialog::txDialog (const txDialog::Layout*)`

Примеры использования:

См. реализацию функции `txInputBox()`.

7.3.10. txDialog (const Layout *layout) [explicit]

Конструктор.

Аргументы:

layout	Макет диалогового окна.

См. также:

`txDialog::Layout, txDialog::setLayout()`

Примеры использования:

См. реализацию функции `txInputBox()`.

7.3.11. Методы

7.3.12. const Layout* setLayout (const Layout *layout)

Устанавливает текущий макет диалогового окна.

Аргументы:

layout	Макет диалогового окна.

Возвращает:

Предыдущий макет.

См. также:

`txDialog::Layout, txDialog::txDialog (const txDialog::Layout*), txDialog::dialogBox()`

Примеры использования:

См. реализацию функции `txInputBox()`.

7.3.13. virtual int dialogProc (HWND_wnd, UINT_msg, WPARAM_wParam, LPARAM_lParam) [virtual]

Функция обработки сообщений диалогового окна.

Аргументы:

_wnd	Дескриптор окна.
_msg	Номер сообщения.
_wParam	1-й параметр сообщения (WORD).
_lParam	2-й параметр сообщения (DWORD).

Возвращает:

В большинстве случаев `false`, детальнее *см.* `DialogProc` в `MSDN`.

Эту функцию надо переопределить для обработки событий окна, или построить её с помощью макросов `TX_BEGIN_MESSAGE_MAP()`, `TX_END_MESSAGE_MAP`, `TX_COMMAND_MAP`.

См. также:

`txDialog::dialogBox()`, `TX_BEGIN_MESSAGE_MAP()`, `TX_END_MESSAGE_MAP`, `TX_COMMAND_MAP`

Примеры использования:

См. реализацию функции `txInputBox()`.

7.3.14. `intptr_t dialogBox (const Layout *layout = NULL, size_tbufsize = 0)`

Запускает диалоговое окно.

Аргументы:

layout	Макет диалогового окна. Если не указан — используется значение, заданное `txDialog::setLayout()` или конструктором `txDialog::txDialog (const txDialog::Layout*)`.

bufsize	Размер буфера для создания шаблона диалога. Если не указан — размер по умолчанию.

Возвращает:

Значение, указанное в функции `EndDialog()`.

По умолчанию — адрес объекта (`this`), для которого была запущена `txDialog::dialogBox()`.

См. также:

`txDialog::dialogProc()`, `txDialog::setLayout()`, `txDialog::Layout`, `txDialog`

Примеры использования:

См. реализацию функции `txInputBox()`.

7.3.15. `intptr_t dialogBox (WORDresource)`

Запускает диалоговое окно.

Аргументы:

resource	Идентификатор диалогового ресурса.

Возвращает:

Значение, указанное в функции `EndDialog()`.

По умолчанию — адрес объекта (`this`), для которого была запущена `txDialog::dialogBox()`.

См. также:

`txDialog::dialogProc()`

Примеры использования:

См. реализацию функции `txInputBox()`.

Объявления и описания членов структуры находятся в файле:`TXLib.h`

7.4. Объединение `txWaveSample_t`

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

7.4.1. Открытые члены

- `operator short * ()`

  Преобразование к массиву значений каналов

7.4.2. Открытые атрибуты

- `short ch [2]`

  Данные каналов: [0] - левый, [1] - правый

7.4.3. Подробное описание

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

Функционально эквивалентен массиву: [0] - левый канал, [1] - правый канал.

См. также:

`txWaveSampleRate`, `txWaveVolMax`, `txWaveData_t`, `txWaveFormat`, `txWaveLoadWav()`, `txWaveOut()`, `txWaveIn()`

Примеры использования:

```
// Сирена 880 Гц, формируемая в буфере

int time = 2000;
txWaveData_t buf (ROUND (time * txWaveSampleRate));

for (unsigned i = 0; i < buf.size(); i++)
      buf[i][0] = buf[i][1] = (short) (sin (i*0.125) * txWaveVolMax);

txWaveOut (buf);    // Проигрываем звук
txWaveOut();        // Ожидаем конца проигрывания
```

8. [TXWAVE] РАБОТА СО ЗВУКОМ

8.1. Классы

- `union txWaveSample_t`

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

8.2. 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()`.

8.3. 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.

8.4. Типы

8.4.1. typedef std::vector<txWaveSample_t> txWaveData_t

Тип, использующийся для буферов данных.

Это `std::vector <txWaveSample_t>`.

txWaveSample_t эквивалентен массиву из двух элементов типа short, соответствующим левому и правому каналу.

Максимальная громкость в канале задаётся константой txWaveVolMax.

См. также:

txWaveData_t, txWaveSampleRate, txWaveVolMax, txWaveFormat, txWaveLoadWav(), txWaveOut(), txWaveIn()

Примеры использования:

```
int time = 2000;
txWaveData_t wav (ROUND (time * txWaveSampleRate));

printf ("Generating the waveform ~880 Hz...\n");

for   (unsigned i = 0; i < wav.size(); i++)
      wav[i][0] = wav[i][1] = (short) (sin (i*0.125) * txWaveVolMax);

printf ("Playing %d sec...\n\n", time/1000);

txWaveOut (buf);     // Проигрываем звук
txWaveOut();         // Ожидаем конца проигрывания
```

8.4.2. typedef bool MonitorProc_t(HWAVEIN waveIn, txWaveData_t &data, void *userData)

Тип функции-монитора для функции txWaveIn().

Аргументы:

waveIn	Дескриптор устройства записи

data	Ссылка на буфер, где находятся записанные данные
userData	Указатель на пользовательские данные monitorData, переданные из функции `txWaveIn()`

Возвращает:

Если надо продолжать запись, то `true`. Если надо закончить запись – `false`.

См. также:

`txWaveIn()`, `MonitorProc()`

Примеры использования:

См. `MonitorProc()`

8.5. Функции

8.5.1. 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.

Аргументы:

timeMs	Время звучания в миллисекундах
freqL	Частота звука в левом канале в Герцах
volL	Громкость звука в левом канале в процентах от максимума
freqR	Частота звука в правом канале в Герцах
volR	Частота звука в правом канале в Герцах

loops	Сколько раз повторять звук
data	Подготовленные данные для воспроизведения (например, загруженные из WAV-файла)

Возвращает:

Дескриптор аудиоустройства воспроизведения

Замечание:

Если во время вызова ещё воспроизводится предыдущий звук, то функция подождёт его завершения, а затем запустит воспроизведение нового.

Параметр `time`:

- Если время звучания `time` положительно, то функция начинает воспроизведение и возвращается, а звук продолжает звучать в фоновом режиме.
- Если отрицательно, то функция будет ожидать завершения воспроизведения звуков в течение |`time`| миллисекунд.
- Вызов функции без параметров будет ожидать окончания воспроизведения всех звуков.
- Если `time` равно нулю, то воспроизведение сразу прекращается.

Гармонические параметры — частоты (`freqL, freqR`) и громкости (`volL, volR`) каналов:

- Если частота левого канала `freqL` не указана или равна `0`, то будут учитываться только данные из буфера левого канала.
- Если громкость левого канала `volL` не указана, то она полагается равной 50%.

- Если частота правого канала `freqR` не указана, то она полагается равной частоте левого канала.
- Если громкость правого канала `volR` не указана, то она полагается равной громкости левого канала.
- Частоты можно сочетать с подготовленными данными буфера `data`.
- Если гармонических данных нет как в левом канале (`freqL == 0` или `volL == 0`), так и в правом (`freqR == 0` или `volR == 0`), то фактическая длительность звучания определяется не параметром `time`, а размером буфера `data`.

Подготовленные данные `data`***:***

- Данные в векторе `data` напрямую влияют на мембраны колонок или наушников.
- На данные в этом векторе не влияют параметры громкости `volL` и `volR`.
- Если вектор `data` не указан, то он не учитывается.
- Подготовленные данные data можно сочетать с частотами `freqL` и `freqR`.
- Вектор `data` должен быть размером не менее, чем `time * txWaveSampleRate`.
- Если размера вектора не хватает для времени воспроизведения в течении времени `time`, то при исчерпании данных вектора они будут игнорироваться.

См. также:

`txWaveIn()`, `txWaveGetPosition()`, `txWaveLoadWav()`, `txWaveSampleRate`, `txWaveVolMax`, `txWaveFormat`, `txWaveData_t`, `txWaveSample_t`

Примеры использования:

```
// Звук 1 сек, 880 Гц, моно, 50% громкости

      txWaveOut (1000, 880); // Запустить проигрывание

// Сирена 880 +/- 88 Гц, моно, 50% громкости

for (double t = -0.1; !_kbhit(); t += 0.1)
      txWaveOut (100, 880 + 88 * cos (t));

// Сирена 880 +/- 88 Гц, стерео

for (double t = -0.1; !_kbhit(); t += 0.1)
      txWaveOut (100, 880 + 88 * cos (t), 100, 880 + 88 * sin (t), 100);
```

8.5.2. HWAVEOUT txWaveOut (const txWaveData_t &data, int loops = 1)

Проигрывает подготовленный или загруженный буфер через звуковую карту.

Аргументы:

data	Подготовленные данные для воспроизведения (например, загруженные из WAV-файла)
loops	Сколько раз повторять звук

Возвращает:

Дескриптор аудиоустройства воспроизведения

См. также:

txWaveIn(), txWaveGetPosition(), txWaveLoadWav(), txWaveSampleRate, txWaveVolMax, txWaveFormat, txWaveData_t, txWaveSample_t

Примеры использования:

```
   // Музыка, загруженная из файла

   txWaveOut (txWaveLoadWav ("Purr.wav"));      // Начинаем проигрываем звук
...
   txSleep();                                   // Ожидаем конца проигрывания

   // Музыка, формируемая в буфере

   int time = 90000;
   txWaveData_t buf (ROUND (time * txWaveSampleRate));
   unsigned size = (unsigned) buf.size();

   for (unsigned i = 0; i < size; i++)
         {
         int t = i/5;
         int val = (((t >> 7) | t | (t >> 6)) * 10 + 4 * ((t & (t >> 13));

         buf[i][0] = buf[i][1] = (short) ((val/127.0 - 1);
         }

   HWAVEOUT waveOut = txWaveOut (buf); assert (waveOut);     // Запускаем звук

   while (txWaveGetPosition (waveOut) < size)          // Проигран ли звук до конца?
         {
         if (_kbhit()) txWaveOut (0);           // Закончить звук, если нажали клавишу
         Sleep (100);
         }
```

8.5.3. bool MonitorProc (HWAVEIN waveIn, txWaveData_t &data, void *userData)

Функция-монитор, регулярно вызывающаяся при записи звука.

Аргументы:

waveIn	Дескриптор устройства записи
Дескриптор устройства записи	Ссылка на буфер, где находятся записанные данные
userData	Указатель на пользовательские данные `monitorData`, переданные из функции `txWaveIn()`

Возвращает:

Если надо продолжать запись, то `true`. Если надо закончить запись — `false`.

См. также:

`txWaveIn()`, `txWaveGetPosition()`, `txWaveVolMax`, `txWaveFormat`, `txWaveData_t`, `txWaveSample_t`

Примеры использования:

```
bool MonitorProc (HWAVEIN, txWaveData_t& data, void*)
      {
      static const char line1[] = "-------------------------";
      static const char line2[] = "|||||||||||||||||||||||||";
      static const int lineSz = sizeof (line1) - 1;

      unsigned pos = (unsigned) data.size();
      unsigned size = (unsigned) data.capacity();
```

```
      unsigned done = (unsigned) (100 * pos/size);

      double avr = 0;
      if    (pos >= 100) for (unsigned i = pos - 100; i < pos; i++)
             avr += (abs (data[i][0]) + abs (data[i][1])) / txWaveVolMax;
      int vol = (avr <= 100)? (int) avr : 100;

printf ("Done %6u/%6u samples [%.*s|%.*s] %3u%% [%.*s%.*s%s %3d%% \r",
      pos, size,
      (int) (lineSz*done/100), line1, (int) (lineSz - lineSz*done/100), line1, done,
      (int) (lineSz*vol /100), line2, (int) (lineSz - lineSz*vol /100), line1, (vol < 100? "]");

return (_kbhit()? (_getch(), false) : true);
}
```

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

Записывает звук со звуковой карты.

Аргументы:

timeMs	Время записи в миллисекундах
monitorProc	Функция, которая вызывается после каждого записанного кадра длиной `frameTime` мсек
monitorData	Указатель на пользовательские данные, которые будут переданы функции `monitorProc`
frameTime	Интервал времени записи, через который будет вызываться функция `monitorProc`, мсек

Возвращает:

Количество записанных данных

Замечания:

Если функция `MonitorProc` не указана, то в процессе записи она не будет вызываться.

Если интервал времени `frameTime` не указан или равен `0`, он устанавливается равным *100 миллисекунд*.

См. также:

`monitorProc()`, `txWaveOut()`, `txWaveGetPosition()`, `txWaveSaveWav()`, `txWaveSampleRate`, `txWaveVolMax`, `txWaveFormat`, `txWaveData_t`, `txWaveSample_t`

Примеры использования:

```
txWaveData_t wav = txWaveIn (20000); // Записываем звук 20 секунд

txWaveOut (wav);     // Проигрываем звук
txWaveOut();         // Ожидаем конца проигрывания

// Пример с функцией-монитором см. в файле TXWaveExample.cpp.
```

8.5.5. unsigned long txWaveGetPosition (void *wave)

Возвращает текущую позицию воспроизведения или записи.

Аргументы:

wave	Устройство воспроизведения или записи

Возвращает:

Количество уже воспроизведенных или уже записанных элементов буферов

См. также:

`txWaveOut()`, `txWaveIn()`, `txWaveSampleRate`, `txWaveFormat`

Примеры использования:

```
HWAVEOUT waveOut = txWaveOut (time, 440); assert (waveOut);    // Запускаем звук

while (txWaveGetPosition (waveOut) < size)          // Проигран ли звук до конца?
      {
      if (_kbhit()) txWaveOut (0);                  // Закончить звук, если нажали клавишу

      printf (".");
      Sleep (100);
      }
```

8.5.6. `txWaveData_t txWaveLoadWav (const char filename[])`

Загружает звуковые данные из WAV-файла.

Аргументы:

filename	Имя файла в формате WAV

Возвращает:

`txWaveData_t`, т.е. `std::vector <txWaveData_t>`, с загруженными данными

Заметки:

Файл должен быть в формате WAV, **Microsoft PCM**, Стерео (2 канала), скорость семплирования 44.100 кГц (`txWaveSampleRate`), 16 бит на канал. Если файл другого формата, его надо преобразовать в нужный формат.

См. также:

`txWaveOut()`, `txWaveIn()`, `txWaveSaveWav()`, `txWaveSampleRate`, `txWaveVolMax`, `txWaveFormat`

Примеры использования:

```
txWaveData_t purr = txWaveLoadWav ("Purr.wav");

txWaveOut (purr);   // Проигрываем звук
txWaveOut();        // Ожидаем конца проигрывания
```

8.5.7. bool txWaveSaveWav (const txWaveData_t &data, const char filename[])

Сохраняет звуковые данные в WAV-файле.

Аргументы:

data	Данные для записи (например, записанные с помощью `txWaveIn()`, или вычисленные по формуле)
filename	Имя файла с расширением WAV

Возвращает:

`true`, если файл открылся на запись, иначе `false`

См. также:

`txWaveIn()`, `txWaveSampleRate`, `txWaveVolMax`, `txWaveFormat`

Примеры использования:

```
txWaveData_t wav = txWaveIn (20000); // Записываем звук
txWaveSaveWav (wav, "Recording.wav"); // Сохраняем звук в файл
```

8.6. Переменные

8.6.1. const double txWaveSampleRate = 44.100

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

См. также:

txWaveFormat, txWaveOut(), txWaveIn(), txWaveGetPosition(), txWaveLoadWav(), txWaveFormat, txWaveSample_t

8.6.2. const WAVEFORMATEX txWaveFormat

Формат аудиоданных для TXWave.

Амплитудная модуляция без сжатия, 2 канала, 16 бит на канал, 44.100 кГц (txWaveSampleRate).

См. также:

txWaveOut(), txWaveIn(), txWaveGetPosition(), txWaveLoadWav(), txWaveSampleRate, txWaveSample_t, txWaveData_t

8.6.3. const double txWaveVolMax

Максимальная громкость в txWaveSample_t, согласно формату txWaveFormat.

См. также:

txWaveSampleRate, txWaveFormat, txWaveData_t, txWaveLoadWav(), txWaveOut(), txWaveIn()

8.7. [TXWave] *Пример для библиотеки TXWave*

Библиотека **TXWave** — воспроизведение и запись звуков

```
//{============================================================================
// Copyright: (C) Ded (Ilya Dedinsky, http://txlib.ru) <mail@txlib.ru>
//}============================================================================

#include "..\TXWave.h"

#include <algorithm>

//----------------------------------------------------------------------------

void BeepDemo();
void SinusDemo();
void StereoDemo();
void Sort1Demo();
void Sort2Demo();
void Sort3Demo();
void BeepDemo2();
void WaveformDemo();
void CatPurrDemo();
void RecordingDemo();
bool MonitorProc (HWAVEIN, txWaveData_t& data, void*);
void Banner (const char* text);
void Pause();

namespace Kernel32 { _TX_DLLIMPORT ("Kernel32", HWND, GetConsoleWindow, (void)); }
```

```
//-----------------------------------------------------------------------------

int main()
	{
	txDisableAutoPause();

	printf ("\n");

	Banner ("1/10: Beep demo");
	BeepDemo();
	Pause();

	txCreateWindow (800, 600);

	Banner ("2/10: Sinus demo");
	SinusDemo();
	Pause();

	printf ("\n" "For idea of the sorting demos see: https://www.youtube.com/watch?v=kPRA0W1kECg\n\n");

	Banner ("3/10: Shaker Sort demo");
	Sort1Demo();
	Pause();

	Banner ("4/10: std::sort (QuickSort) demo");
```

```
    Sort2Demo();
    Pause();

    Banner ("5/10: std::stable_sort (MergeSort) demo");
    Sort3Demo();
    Pause();

    ShowWindow (Kernel32::GetConsoleWindow(), SW_SHOW);
    ShowWindow (txWindow(), SW_HIDE);

    Banner ("6/10: Stereo demo");
    StereoDemo();
    Pause();

    Banner ("7/10: Beep demo 2");
    BeepDemo2();
    Pause();

    Banner ("8/10: Waveform demo");
    WaveformDemo();
    Pause();

    Banner ("9/10: Cat purr demo");
    CatPurrDemo();
    Pause();
```

```
      Banner ("10/10: Recording demo");
      RecordingDemo();
      Pause();

      Banner ("All demos done");
      return 0;
}

//---------------------------------------------------------------------------

void BeepDemo()
      {
      txWaveOut (250, 880);
      }

//---------------------------------------------------------------------------

void SinusDemo()
      {
      POINT size = txGetExtent();

      double x0 = -2*M_PI, x1 = +2*M_PI;
      double zoom = 50;

      for (int i = -10; i <= 10; i++)
            {
```

```
            txSetColor (i? TX_DARKGRAY : TX_WHITE);

            txLine (0, size.y/2 + i*zoom, size.x, size.y/2 + i*zoom);
            txLine (size.x/2 + i*zoom, 0, size.x/2 + i*zoom, size.y);
            }

      Banner ("\f" "SinusDemo. Press any key to stop...");

      for (double x = x0; x <= x1 && !_kbhit(); x += (x1 - x0) / 101)
            {
            double y = 5 * sin (x);

            txWaveOut (50, 880 + 88 * y);

            txCircle (x*zoom + size.x/2, -y*zoom + size.y/2, 5);
            }
      }

//---------------------------------------------------------------------------

void StereoDemo()
      {
      for (double t = -0.1; !_kbhit(); t += 0.1)
            {
            txWaveOut (50, 880 + 88 * cos (t), 100,
                           880 + 88 * sin (t), 100);
```

```
            const char line[] = "----------------------------------------";
            const int lineSz = sizeof (line) - 1;

            int sz = (int) ROUND ((1 + cos (t))/2 * lineSz);
            printf ("Press any key to stop... [%.*s|%.*s] \r", lineSz-sz, line, sz, line);
            }

      printf ("\n\n");
      }

//--------------------------------------------------------------------------------

void BeepDemo2()
      {
      int time = 2000;
      txWaveData_t wav (ROUND (time * txWaveSampleRate));

      printf ("Generating the waveform ~880 Hz...\n");

      for (unsigned i = 0; i < wav.size(); i++)
            wav[i][0] = wav[i][1] = (short) (sin (i*0.125) * txWaveVolMax); // * 2*M_PI/44100 * 880

      printf ("Playing %d sec...\n\n", time/1000);

      txWaveOut (wav);
```

```
      txWaveOut();
}

//---------------------------------------------------------------------------

void WaveformDemo()
      {
      int time = 90000;
      txWaveData_t buf (ROUND (time * txWaveSampleRate));
      unsigned size = (unsigned) buf.size();

      printf ("Generating the wave: (t >> 7 | t | t >> 6) * 10 + 4 * (t & t >> 13 | t >> 6)...\n"
            "See: http://countercomplex.blogspot.ru/2011/10/algorithmic-symphonies-from-one-
line-of.html\n\n");

      for (unsigned i = 0; i < size; i++)
            {
      int t = i/5;
      int val = (((t >> 7) | t | (t >> 6)) * 10 + 4 * ((t & (t >> 13)) | (t >> 6))) & 0xFF;

      buf[i][0] = buf[i][1] = (short) ((val/127.0 - 1) / (1 + pow (15, 17.0*i/size - 15)) *
txWaveVolMax);
      }

      printf ("Now playing, press a key to stop...\n\n");
```

```
	HWAVEOUT waveOut = txWaveOut (buf); assert (waveOut);

	while (!_kbhit())
		{
		unsigned long pos = txWaveGetPosition (waveOut);
		unsigned done = (unsigned) (100 * pos/size);

		const char line[] = "----------------------------------------";
		const int lineSz = sizeof (line) - 1;

		printf ("Done %7lu/%7u samples [%.*s|%.*s] %3u%% \r", pos, size, (int)
(lineSz*done/100), line, (int) (lineSz - lineSz*done/100), line, done);

		if (done >= 100) break;
		Sleep (100);
		}

	txWaveOut (0);
	printf ("\n\n");
	}

//------------------------------------------------------------------------------

void CatPurrDemo()
	{
	txWaveData_t purr = txWaveLoadWav ("Purr.wav");
```

```
	txWaveOut (purr);

	printf ("Uninterruptable... do relax and wait %ld sec\n\n", ROUND ((int) purr.size() /
txWaveSampleRate / 1000));

	txWaveOut();
	}

//---------------------------------------------------------------------------

void RecordingDemo()
	{
	Banner ("\r" "Turn on the microphone and get prepared for a recording."); Pause();

	int time = 20000;

	printf ("Recording %2d sec...\n", time/1000);

	txWaveData_t wav = txWaveIn (time, MonitorProc);
	time = ROUND ((int) wav.size() / txWaveSampleRate);

	printf ("\nRecorded: %2d sec\n", time/1000);
	printf ("\nPlaying %2d sec...\n", time/1000);

	HWAVEOUT waveOut = txWaveOut (wav);
```

```
      while (!_kbhit() && txWaveGetPosition (waveOut) < wav.size()) Sleep (100);
      txWaveOut (0);

      printf ("\nSaving to WAV file...\n\n");
 txWaveSaveWav (wav, "Recording.wav");
      }

bool MonitorProc (HWAVEIN, txWaveData_t& data, void*)
      {
      static const char line1[] = "----------------";
      static const char line2[] = "||||||||||||||||";
      static const int lineSz = sizeof (line1) - 1;

      unsigned pos = (unsigned) data.size();
      unsigned size = (unsigned) data.capacity();
      unsigned done = (unsigned) (100 * pos/size);

      double avr = 10;
      if (pos >= 100) for (unsigned i = pos - 100; i < pos; i++) avr += (abs (data[i][0]) +
abs (data[i][1])) / txWaveVolMax;
      int vol = (avr <= 100)? (int) avr : 100;

      printf ("Done %6u/%6u samples [%.*s|%.*s] %3u%% [%.*s%.*s%s %3d%% \r",
              pos, size,
             (int) (lineSz*done/100), line1, (int) (lineSz - lineSz*done/100), line1, done,
             (int) (lineSz*vol /100), line2, (int) (lineSz - lineSz*vol /100), line1, (vol < 100?
```

```
"]" : "X"), vol);

      return (_kbhit()? ((void) _getch(), false) : true);
      }

//---------------------------------------------------------------------------

// For idea of this demo see: https://www.youtube.com/watch?v=kPRA0W1kECg

struct Number_t
      {
      int val_;

      Number_t();

      Number_t& operator = (const Number_t& that);
      bool operator < (const Number_t& that) const;

      bool draw (COLORREF color, COLORREF fillColor) const;

      static void setinfo (const Number_t* data, size_t size);

      static const Number_t* data__;
      static size_t size__;
}     ;
```

```
const Number_t* Number_t::data__ = NULL;
      size_t Number_t::size__ = 0;

Number_t::Number_t() :
      val_ (88 + rand() % 2552)
      {}

void Number_t::setinfo (const Number_t* data, size_t size)
      {
      data__ = data;
      size__ = size;

      for (unsigned i = 0; i < size__; i++) data__[i].draw (RGB (64, 64, 64), RGB (32, 32, 32));
      }

bool Number_t::draw (COLORREF color, COLORREF fillColor) const
      {
      if (_kbhit()) throw EOF;

      const int width = 9;

      int n = (int) (this - data__);
      if (!data__ || !(0 <= n && n < (int) size__)) return false;

      int x0 = txGetExtentX()/2 - width * (int)size__/2;
```

```
        txSetColor (TX_BLACK); txSetFillColor (TX_BLACK);
        txRectangle (x0 + n*width, 500, x0 + (n+1)*width, 500 - 400);

        txSetColor (color); txSetFillColor (fillColor);
        txRectangle (x0 + n*width, 500, x0 + (n+1)*width, 500 - val_/10);

        return true;
        }

Number_t& Number_t::operator = (const Number_t& that)
        {
        if (_kbhit()) throw EOF;

        val_ = that.val_;
        draw (TX_WHITE, TX_LIGHTGRAY);

        return *this;
        }

bool Number_t::operator < (const Number_t& that) const
        {
        if (_kbhit()) throw EOF;

        this->draw (TX_LIGHTRED, TX_RED);
        that. draw (TX_LIGHTRED, TX_RED);
```

```
      txWaveOut (50, val_, 10, that.val_, 10);

      this->draw (TX_WHITE, TX_LIGHTGRAY);
      that. draw (TX_WHITE, TX_LIGHTGRAY);

      return val_ < that.val_;
      }

void Sort1Demo()
      {
      try
            {
            Number_t data[16];
            Number_t::setinfo (data, sizearr (data));

            for (int left = 0, right = (int) sizearr (data) - 1; left < right; )
                {
            bool change = false;

            for (int i = left; i < right; i++)
                  if (data[i+1] < data[i]) { std::swap (data[i+1], data[i]); change = true; }

            right--;
            if (!change) break;

            change = false;
```

```
            for (int i = right-1; i >= left; i--)
                  if (data[i+1] < data[i]) { std::swap (data[i+1], data[i]); change = true; }

            left++;
            if (!change) break;
            }
      }

      catch (...) {}
      }

void Sort2Demo()
      {
      try
            {
            Number_t data[32];
            Number_t::setinfo (data, sizearr (data));

            std::sort (data, data + sizearr (data));
            }

      catch (...) {}
      }

void Sort3Demo()
```

```
	{
	try
		{
		Number_t data[32];
		Number_t::setinfo (data, sizearr (data));

		std::stable_sort (data, data + sizearr (data));
		}

	catch (...) {}
	}

//-----------------------------------------------------------------------------

void Banner (const char* text)
	{
	if (txWindow() && IsWindowVisible (txWindow()))
		{
		static POINT size = txGetExtent();

		if (*text != '\f')
			{
			txSetFillColor (TX_BLACK);
			txClear();
			}
		else
```

```
                text++;

        if (*text == '\r') text++;
        if (*text == '\t') text++;

        txSetFillColor (TX_WHITE);
        txSetTextAlign (TA_CENTER);

        txSetColor (TX_WHITE); txRectangle (0, size.y * 0.95, size.x, size.y);
        txSetColor (TX_BLACK); txDrawText (0, size.y * 0.95, size.x, size.y, text);
        txSleep (0);

        txSetColor (TX_WHITE);
        }
    else
        {
        y;
        if (*text == '\f') { text++; }
        if (*text == '\r') { r; text++; }
        if (*text == '\t') { d; text++; }

        printf ("%s\n\n", text);

        d;
        }
    }
```

```
void Pause()
      {
      Banner ("\f\t" "Press a key to continue...");

      while (_kbhit()) (void) _getch();
      (void) _getch();
      }
```

9. ПРИМЕРЫ

9.1. Пример: Улучшенный

```
//{===========================================================================
// Copyright: (C) Ded (Ilya Dedinsky, http://txlib.ru) <mail@txlib.ru>
//}===========================================================================

#include "TXLib.h"

int main()
      {
      txCreateWindow (800, 600);

      txSetColor (TX_WHITE);
      txSetFillColor (TX_TRANSPARENT);
      txRectangle (10, 10, 790, 590);
```

```
    txSetColor (TX_LIGHTCYAN);
    txEllipse (200, 150, 600, 450);
    txEllipse (245, 150, 555, 450);
    txEllipse (290, 150, 510, 450);
    txEllipse (330, 150, 470, 450);
    txEllipse (375, 150, 425, 450);
    txEllipse (200, 150, 600, 450);
    txEllipse (200, 190, 600, 410);
    txEllipse (200, 230, 600, 370);
    txEllipse (200, 270, 600, 330);
    txLine (200, 300, 600, 300);

    txSetColor (TX_LIGHTGREEN);
    txSelectFont ("Times New Roman", 60);
    txSetTextAlign (TA_CENTER);
    txTextOut (400, 480, "Hello, world!");

    txSetColor (TX_YELLOW);
    txSetFillColor (TX_YELLOW);
    txLine (385, 135, 385, 120);
    txLine (385, 135, 375, 150);
    txLine (385, 135, 395, 150);
    txLine (385, 125, 375, 135);
    txLine (385, 125, 400, 120);
    txCircle (385, 115, 6);
```

```
    txSetFillColor (TX_TRANSPARENT);
    txLine (400, 75, 400, 150);
    txRectangle (400, 75, 450, 115);
    txSelectFont ("Times New Roman", 20);
    txTextOut (425, 85, "C++");

    txTextCursor (false);
    return 0;
    }
```

9.2. Пример: Функции

```
//{============================================================================
// Copyright: (C) Ded (Ilya Dedinsky, http://txlib.ru) <mail@txlib.ru>
//}============================================================================

#include "TXLib.h"

//----------------------------------------------------------------------------

void DrawMan();
void DrawEarth();
void DrawFlag();
void DrawHello();
void DrawFrame();

//----------------------------------------------------------------------------
```

```
int main()
      {
      txCreateWindow (800, 600);

      DrawFrame();
      DrawEarth();
      DrawHello();
      DrawMan();
      DrawFlag();

      txTextCursor (false);
      return 0;
      }

//---------------------------------------------------------------------------

void DrawEarth()
      {
      txSetColor (TX_LIGHTCYAN);
      txEllipse (200, 150, 600, 450);
      txEllipse (245, 150, 555, 450);
      txEllipse (290, 150, 510, 450);
      txEllipse (330, 150, 470, 450);
      txEllipse (375, 150, 425, 450);
      txEllipse (200, 150, 600, 450);
      txEllipse (200, 190, 600, 410);
      txEllipse (200, 230, 600, 370);
```

```
      txEllipse (200, 270, 600, 330);
      txLine    (200, 300, 600, 300);
      }

//---------------------------------------------------------------------------
void DrawMan()
      {
      txSetColor (TX_YELLOW);
      txSetFillColor (TX_YELLOW);
      txLine   (385, 135, 385, 120);
      txLine   (385, 135, 375, 150);
      txLine   (385, 135, 395, 150);
      txLine   (385, 125, 375, 135);
      txLine   (385, 125, 400, 120);
      txCircle (385, 115, 6);
      }

//---------------------------------------------------------------------------
void DrawFlag()
      {
      txSetFillColor (TX_TRANSPARENT);
      txLine      (400, 75, 400, 150);
      txRectangle (400, 75, 450, 115);
      txSelectFont ("Times New Roman", 20);
      txTextOut    (425, 85, "C++");
      }
```

```
//---------------------------------------------------------------------------

void DrawHello()
      {
      txSetColor (TX_LIGHTGREEN);
      txSelectFont ("Times New Roman", 60);
      txSetTextAlign (TA_CENTER);
      txTextOut (400, 480, "Hello, world!");
      }

//---------------------------------------------------------------------------

void DrawFrame()
      {
      txSetColor (TX_WHITE);
      txSetFillColor (TX_TRANSPARENT);
      txRectangle (10, 10, 790, 590);
      }
```

9.3. Пример: Функции с параметрами

```
//{===========================================================================
// Copyright: (C) Ded (Ilya Dedinsky, http://txlib.ru) <mail@txlib.ru>
//}===========================================================================

#include "TXLib.h"
```

```
//---------------------------------------------------------------------------

void DrawMan       (int x, int y, int sizeX, int sizeY, COLORREF color,
                    double hand = 0, double legs = 0, double head = 0, double twist = 0);
void DrawEarth     (int x, int y, int sizeX, int sizeY, COLORREF color);
void DrawFlag      (int x, int y, int sizeX, int sizeY, COLORREF color, COLORREF bkColor);
void DrawHello     (int x, int y, const char* text, int size, COLORREF color);
void DrawFrame     (int sizeX, int sizeY, int size, COLORREF color);

//---------------------------------------------------------------------------

int main()
      {
      txCreateWindow (800, 600);

      DrawFrame (800, 600, 10, TX_WHITE);
      DrawHello (400, 480, "Hello, world!", 60, TX_LIGHTGREEN);

      DrawEarth (400, 300, 400, 300, TX_LIGHTCYAN);

      DrawFlag (400, 150, 50, 75, TX_YELLOW, TX_TRANSPARENT);
      DrawMan (385, 150, 20, 40, TX_YELLOW, 0, 0, 0);

      txTextCursor (false);
      return 0;
      }
```

```
//--------------------------------------------------------------------------

void DrawMan       (int x, int y, int sizeX, int sizeY, COLORREF color,
                    double hand, double legs, double head, double twist)
      {
      txSetColor (color);
      txSetFillColor (color);

      txLine (x, y - (0.35 + twist) * sizeY, x, y - 0.7*sizeY);

      txLine (x, y - (0.35 + twist) * sizeY, x - (0.5 + legs) * sizeX, y);
      txLine (x, y - (0.35 + twist) * sizeY, x + (0.5 + legs) * sizeX, y);

      txLine (x, y - 0.65*sizeY, x - sizeX/2, y - 0.4*sizeY);
      txLine (x, y - 0.65*sizeY, x + sizeX/1.2, y - (0.7 + hand) * sizeY);

      txCircle (x, y - sizeY + (0.3 + head) * sizeX, 0.3*sizeX);
      }

//--------------------------------------------------------------------------

void DrawEarth (int x, int y, int sizeX, int sizeY, COLORREF color)
      {
      txSetColor (color);

      int r = sizeX/2;
```

```
      while (r >= 0)
            {
            txEllipse (x-r, y-sizeY/2, x+r, y+sizeY/2);
            r -= sizeX/9;
            }

      r = sizeY/2;
      while (r >= 0)
            {
            txEllipse (x-sizeX/2, y-r, x+sizeX/2, y+r);
            r -= sizeY/6;
            }

      txLine (x - sizeX/2, y, x + sizeX/2, y);
      }

//---------------------------------------------------------------------------

void DrawFlag (int x, int y, int sizeX, int sizeY, COLORREF color, COLORREF bkColor)
      {
      txSetColor (color);
      txSetFillColor (bkColor);

      txLine (x, y, x, y - sizeY);
      txRectangle (x, y - sizeY/2, x + sizeX, y - sizeY);
```

```
      txSelectFont ("Times New Roman", 20);
      txTextOut (x + sizeX/2, y - sizeY*7/8, "C++");
      }

//---------------------------------------------------------------------------

void DrawHello (int x, int y, const char* text, int size, COLORREF color)
      {
      txSetColor (color);

      txSelectFont ("Times New Roman", size);
      txSetTextAlign (TA_CENTER);

      txTextOut (x, y, text);
      }

//---------------------------------------------------------------------------

      void DrawFrame (int sizeX, int sizeY, int size, COLORREF color)
      {
      txSetColor (color);
      txSetFillColor (TX_TRANSPARENT);

      txRectangle (size, size, sizeX-size, sizeY-size);
      }
```

9.4. Пример: Циклы

```
//{============================================================================
// Copyright: (C) Ded (Ilya Dedinsky, http://txlib.ru) <mail@txlib.ru>
//}============================================================================

#include <conio.h>
#include "TXLib.h"

//----------------------------------------------------------------------------

void JumpMan       (int x, int y, int sizeX, int sizeY, double jump,
                    COLORREF color, COLORREF bkColor, int jumps, int delay);

void MoveMan       (int fromX, int fromY, int toX, int toY,
                    int sizeX, int sizeY, COLORREF color, COLORREF bkColor,
                    int time, int steps);

void DrawMan       (int x, int y, int sizeX, int sizeY, COLORREF color,
                    double hand = 0, double legs = 0, double head = 0, double twist = 0);

void AppearEarth   (int x, int y, int sizeX, int sizeY, COLORREF from, COLORREF to,
                    int time, int steps);

void DrawEarth     (int x, int y, int sizeX, int sizeY, COLORREF color);
```

```
void AppearText    (int x, int y, const char* text, COLORREF from, COLORREF to,
                    int time, int steps);

void DrawHello     (int x, int y, const char* text, int size, COLORREF color);

void UnwindFlag    (int x, int y, int fromSizeX, int toSizeX, int sizeY,
                   COLORREF color, COLORREF bkColor, int time, int steps);

void DrawFlag      (int x, int y, int sizeX, int sizeY, COLORREF color, COLORREF bkColor);

void DrawFrame     (int sizeX, int sizeY, int size, COLORREF color);

//---------------------------------------------------------------------------

int main()
      {
      txCreateWindow (800, 600);
      txTextCursor (false);

      DrawFrame (800, 600, 10, TX_WHITE);

      txSelectFont ("Times New Roman", 60);
      txSetTextAlign (TA_CENTER);

      AppearText  (400, 480, "Hello, world!",      TX_BLACK, TX_LIGHTGREEN, 5000, 100);
      AppearEarth (400, 300, 400, 300,             TX_BLACK, TX_LIGHTCYAN, 5000, 100);
```

```
      MoveMan     (20, 150, 385, 150, 20, 40,     TX_YELLOW, TX_BLACK, 3000, 100);
      JumpMan     (385, 150, 20, 40, 0.25,        TX_YELLOW, TX_BLACK, 10, 100);
      UnwindFlag  (400, 150, 0, 40, 60,           TX_YELLOW, TX_BLACK, 500, 100);

      return 0;
      }

//---------------------------------------------------------------------------

void JumpMan (int x, int y, int sizeX, int sizeY, double jump,
COLORREF color, COLORREF bkColor, int jumps, int delay)
      {
      DrawMan (x, y, sizeX, sizeY, TX_BLACK, 0, 0, 0, 0);

      txBegin();

      int i = 0;
      while (i < jumps)
            {
            DrawMan (x, y - (int)(i%2 * jump*10), sizeX, sizeY, color,
                    (i%2-0.5) * jump/3, 0, (i%2-0.5) * -jump/3, 0);

            txSleep (delay);

            DrawMan (x, y - (int)(i%2 * jump*10), sizeX, sizeY, bkColor,
```

```
                    (i%2-0.5) * jump/3, 0, (i%2-0.5) * -jump/3, 0);
            i++;
            }

      DrawMan (x, y - (int)(jumps%2 * jump*10), sizeX, sizeY, color,
              (jumps%2-0.5) * jump/3, 0, (jumps%2-0.5) * -jump/3, 0);

      txEnd();
      }

//---------------------------------------------------------------------------

void MoveMan       (int fromX, int fromY, int toX, int toY,
                    int sizeX, int sizeY, COLORREF color, COLORREF bkColor,
                     int time, int steps)
      {
      txBegin();

      int i = 0;
      while (i <= steps)
            {
            int x = fromX + (toX - fromX) * i/steps,
                y = fromY + (toY - fromY) * i/steps;

            DrawMan (x, y - i%6, sizeX, sizeY, color, i%3*0.02, i%3*-0.1, i%3*0.1, 0);
```

```
            txSleep (time / steps);

            DrawMan (x, y - i%6, sizeX, sizeY, bkColor, i%3*0.02, i%3*-0.1, i%3*0.1, 0);
            i++;
            }

      DrawMan (toX, toY, sizeX, sizeY, color, 0, 0, 0);

      txEnd();
      }

//---------------------------------------------------------------------------

void UnwindFlag    (int x, int y, int fromSizeX, int toSizeX, int sizeY,
                    COLORREF color, COLORREF bkColor, int time, int steps)
      {
      txBegin();

      int i = 0;
      while (i <= steps)
            {
            int sizeX = fromSizeX + (toSizeX - fromSizeX) * i/steps;

            DrawFlag (x, y, sizeX, sizeY, color, bkColor);

            txSleep (time / steps);
```

```
            DrawFlag (x, y, sizeX, sizeY, bkColor, bkColor);
            i++;
            }

      DrawFlag (x, y, toSizeX, sizeY, color, bkColor);

      txEnd();
      }

//---------------------------------------------------------------------------

void DrawMan (int x, int y, int sizeX, int sizeY, COLORREF color,
              double hand, double legs, double head, double twist)
      {
      txSetColor (color);
      txSetFillColor (color);

      txLine (x + twist*sizeX, y - 0.35*sizeY, x, y - 0.7*sizeY);

      txLine (x + twist*sizeX, y - 0.35*sizeY, x - (0.5 + legs) * sizeX, y);
      txLine (x + twist*sizeX, y - 0.35*sizeY, x + (0.5 + legs) * sizeX, y);

      txLine (x, y - 0.65*sizeY, x - sizeX/2,   y - 0.4*sizeY);
      txLine (x, y - 0.65*sizeY, x + sizeX/1.2, y - (0.7 + hand) * sizeY);
```

```
      txCircle (x, y - sizeY + (0.3 + head) * sizeX, 0.3*sizeX);
      }

//---------------------------------------------------------------------------

void DrawEarth (int x, int y, int sizeX, int sizeY, COLORREF color)
      {
      txSetColor (color);

      int r = sizeX/2;
      while (r >= 0)
            {
            txEllipse (x-r, y-sizeY/2, x+r, y+sizeY/2);
            r -= sizeX/9;
            }

      r = sizeY/2;
      while (r >= 0)
            {
            txEllipse (x-sizeX/2, y-r, x+sizeX/2, y+r);
            r -= sizeY/6;
            }

      txLine (x - sizeX/2, y, x + sizeX/2, y);
      }
```

```
//---------------------------------------------------------------------------

void AppearEarth (int x, int y, int sizeX, int sizeY, COLORREF from, COLORREF to,
                  int time, int steps)
      {
      int   r0 = txExtractColor (from, TX_RED),   r1 = txExtractColor (to, TX_RED),
            g0 = txExtractColor (from, TX_GREEN), g1 = txExtractColor (to, TX_GREEN),
            b0 = txExtractColor (from, TX_BLUE),  b1 = txExtractColor (to, TX_BLUE);

      int i = 0;
      while (i <= steps)
            {
            int   r = r0 + (r1 - r0) * i/steps,
                  g = g0 + (g1 - g0) * i/steps,
                  b = b0 + (b1 - b0) * i/steps;

            DrawEarth (x, y, sizeX, sizeY, RGB (r, g, b));

            Sleep (time / steps);
            i++;
            }
      }

//---------------------------------------------------------------------------
```

```
void AppearText    (int x, int y, const char* text, COLORREF from, COLORREF to,
                    int time, int steps)
      {
      int    r0 = txExtractColor (from, TX_RED),    r1 = txExtractColor (to, TX_RED),
             g0 = txExtractColor (from, TX_GREEN),  g1 = txExtractColor (to, TX_GREEN),
             b0 = txExtractColor (from, TX_BLUE),   b1 = txExtractColor (to, TX_BLUE);

      for (int i = 0; i <= steps; i++)
             {
             int    r = r0 + (r1 - r0) * i/steps,
                    g = g0 + (g1 - g0) * i/steps,
                    b = b0 + (b1 - b0) * i/steps;

             txSetColor (RGB (r, g, b));
             txTextOut (x, y, text);

             Sleep (time / steps);
             }
      }

//---------------------------------------------------------------------------

void DrawFlag (int x, int y, int sizeX, int sizeY, COLORREF color, COLORREF bkColor)
      {
      txSetColor (color);
```

```
      txSetFillColor (bkColor);

      txLine (x, y, x, y - sizeY);
      txRectangle (x, y - sizeY/2, x + sizeX, y - sizeY);

      txSelectFont ("Times New Roman", sizeX/2 + 1);
      txTextOut (x + sizeX/2, y - sizeY*7/8, "C++");
      }
//---------------------------------------------------------------------------
void DrawHello (int x, int y, const char* text, int size, COLORREF color)
      {
      txSetColor (color);

      txSelectFont ("Times New Roman", size);
      txSetTextAlign (TA_CENTER);

      txTextOut (x, y, text);
      }
//---------------------------------------------------------------------------
void DrawFrame (int sizeX, int sizeY, int size, COLORREF color)
      {
      txSetColor (color);
      txSetFillColor (TX_TRANSPARENT);

      txRectangle (size, size, sizeX-size, sizeY-size);
      }
```

9.5. Пример: Циклы (2)

```
//{===========================================================================
// Copyright: (C) Ded (Ilya Dedinsky, http://txlib.ru) <mail@txlib.ru>
//}===========================================================================

// Press Alt+Ctrl+Shift+F12 to view this info in About Box

#define __MODULE "Example06"
#define __VERSION "1.0"
#define __AUTHOR "Ilya Dedinsky"
#define __DESCRIPTION "Пример использования библиотеки TXLib"
//---------------------------------------------------------------------------

#include <conio.h>
#include "TXLib.h"
//---------------------------------------------------------------------------

void DanceMan (int x, int y, int sizeX, int sizeY, double jump,
COLORREF color, COLORREF bkColor, int delay);

void JumpMan (int x, int y, int sizeX, int sizeY, double jump,
COLORREF color, COLORREF bkColor, int jumps, int delay);

void MoveMan (int fromX, int fromY, int toX, int toY,
int sizeX, int sizeY, COLORREF color, COLORREF bkColor,
int time, int steps);
```

```
void DrawMan (int x, int y, int sizeX, int sizeY, COLORREF color,
double hand = 0, double legs = 0, double head = 0, double twist = 0);

void AppearEarth (int x, int y, int sizeX, int sizeY, COLORREF from, COLORREF to,
int time, int steps);

void DrawEarth (int x, int y, int sizeX, int sizeY, COLORREF color);

void AppearText (int x, int y, const char* text, COLORREF from, COLORREF to,
int time, int steps);

void DrawHello (int x, int y, const char* text, int size, COLORREF color);

void UnwindFlag (int x, int y, int fromSizeX, int toSizeX, int sizeY,
COLORREF color, COLORREF bkColor, int time, int steps);

void DrawFlag (int x, int y, int sizeX, int sizeY, COLORREF color, COLORREF bkColor);

void DrawFrame (int sizeX, int sizeY, int size, COLORREF color);

int kbget();

//---------------------------------------------------------------------------

int main()
      {
```

```
      txCreateWindow (800, 600);
      txTextCursor (false);

      int sizeX = txGetExtentX(), sizeY = txGetExtentY();
      int centerX = (sizeX+1)/2, centerY = (sizeY+1)/2;

      DrawFrame (sizeX, sizeY, 10, TX_WHITE);

      txSelectFont ("Times New Roman", 60);
      txSetTextAlign (TA_CENTER);

      AppearText (centerX, sizeY*4/5, "\"Hello, world!\\n\" :)",
TX_BLACK, TX_LIGHTGREEN, 5000, 100);
      AppearEarth (centerX, centerY, sizeX/2, sizeY/2,
TX_BLACK, TX_LIGHTCYAN,  5000, 100);

      MoveMan (20, centerY - sizeY/4, centerX - sizeX/50, centerY - sizeY/4, sizeX/40, sizeY/15,
TX_YELLOW, TX_BLACK, 3000, 100);
      txSleep (150);
      JumpMan (centerX - sizeX/50, centerY - sizeY/4, sizeX/40, sizeY/15, 0.25,
TX_YELLOW, TX_BLACK, 10, 100);

      txPlaySound ("tada.wav"); // Windows < 7
      txPlaySound ("C:\\Windows\\Media\\tada.wav"); // Windows >= 7

      UnwindFlag (centerX, centerY - sizeY/4, 0, sizeX/20, sizeY/10,
```

```
TX_YELLOW, TX_BLACK, 500, 100);

      DanceMan (centerX - sizeX/50, centerY - sizeY/4, sizeX/40, sizeY/15, 0.25,
TX_YELLOW, TX_BLACK, 200);

      return 0;
      }
//---------------------------------------------------------------------------
void JumpMan (int x, int y, int sizeX, int sizeY, double jump,
              COLORREF color, COLORREF bkColor, int jumps, int delay)
      {
      DrawMan (x, y, sizeX, sizeY, TX_BLACK, 0, 0, 0, 0);

      txBegin();

      for (int i = 0; i < jumps && !_kbhit(); i++)
            {
            DrawMan (x, y - (int)(i%2 * jump*10), sizeX, sizeY, color,
                    (i%2-0.5) * jump/3, 0, (i%2-0.5) * -jump/3, 0);

            txSleep (delay);

            DrawMan (x, y - (int)(i%2 * jump*10), sizeX, sizeY, bkColor,
                    (i%2-0.5) * jump/3, 0, (i%2-0.5) * -jump/3, 0);
            }
```

```
      kbget();

      DrawMan (x, y - (int)(jumps%2 * jump*10), sizeX, sizeY, color,
              (jumps%2-0.5) * jump/3, 0, (jumps%2-0.5) * -jump/3, 0);

      txEnd();
      }

//---------------------------------------------------------------------------

void DanceMan (int x, int y, int sizeX, int sizeY, double jump,
               COLORREF color, COLORREF bkColor, int delay)
      {
      DrawMan (x, y, sizeX, sizeY, TX_BLACK, 0, 0, 0, 0);

      txBegin();

      int i = 0;
      for (; !_kbhit(); i++)
            {
            DrawMan (x, y - (int)(i%2 * jump*10), sizeX, sizeY, color,
            (i%2-0.5) * jump/3, 0, (i%2-0.5) * -jump/3, (i%2-0.5) * jump);

            txSleep (delay);

            DrawMan (x, y - (int)(i%2 * jump*10), sizeX, sizeY, bkColor,
```

```
                    (i%2-0.5) * jump/3, 0, (i%2-0.5) * -jump/3, (i%2-0.5) * jump);
            }

      kbget();

      DrawMan (x, y - (int)(i%2 * jump*10), sizeX, sizeY, color,
              (i%2-0.5) * jump/3, 0, (i%2-0.5) * -jump/3, (i%2-0.5) * jump);

      txEnd();
      }

//---------------------------------------------------------------------------

void MoveMan (int fromX, int fromY, int toX, int toY,
              int sizeX, int sizeY, COLORREF color, COLORREF bkColor,
              int time, int steps)
      {
      txBegin();

      for   (int i = 0; i <= steps && !_kbhit(); i++)
            {
            int x = fromX + (toX - fromX) * i/steps,
                y = fromY + (toY - fromY) * i/steps;

            DrawMan (x, y - i%6, sizeX, sizeY, color, i%3*0.02, i%3*-0.1, i%3*0.1, 0);
```

```
            txSleep (time / steps);

            DrawMan (x, y - i%6, sizeX, sizeY, bkColor, i%3*0.02, i%3*-0.1, i%3*0.1, 0);
            }

      kbget();

      DrawMan (toX, toY, sizeX, sizeY, color, 0, 0, 0);

      txEnd();
      }

//---------------------------------------------------------------------------

void UnwindFlag    (int x, int y, int fromSizeX, int toSizeX, int sizeY,
                    COLORREF color, COLORREF bkColor, int time, int steps)
      {
      txBegin();

      for   (int i = 0; i <= steps && !_kbhit(); i++)
            {
            int sizeX = fromSizeX + (toSizeX - fromSizeX) * i/steps;

            DrawFlag (x, y, sizeX, sizeY, color, bkColor);

            txSleep (time / steps);
```

```
            DrawFlag (x, y, sizeX, sizeY, bkColor, bkColor);
            }

      kbget();

      DrawFlag (x, y, toSizeX, sizeY, color, bkColor);

      txEnd();
      }
//---------------------------------------------------------------------------
void DrawMan (int x, int y, int sizeX, int sizeY, COLORREF color,
              double hand, double legs, double head, double twist)
      {
      txSetColor (color);
      txSetFillColor (color);

      txLine (x + twist*sizeX, y - 0.35*sizeY, x, y - 0.7*sizeY);

      txLine (x + twist*sizeX, y - 0.35*sizeY, x - (0.5 + legs) * sizeX, y);
      txLine (x + twist*sizeX, y - 0.35*sizeY, x + (0.5 + legs) * sizeX, y);

      txLine (x, y - 0.65*sizeY, x - sizeX/2, y - 0.4*sizeY);
      txLine (x, y - 0.65*sizeY, x + sizeX/1.2, y - (0.7 + hand) * sizeY);

      txCircle (x, y - sizeY + (0.3 + head) * sizeX, 0.3*sizeX);
      }
```

```
//--------------------------------------------------------------------------------

void DrawEarth (int x, int y, int sizeX, int sizeY, COLORREF color)
      {
      txSetColor (color);
      for (int rx = sizeX/2; rx >= 0; rx -= sizeX/9) txEllipse (x-rx, y-sizeY/2, x+rx, y+sizeY/2);
      for (int ry = sizeY/2; ry >= 0; ry -= sizeY/6) txEllipse (x-sizeX/2, y-ry, x+sizeX/2, y+ry);
      txLine (x - sizeX/2, y, x + sizeX/2, y);
      }
//--------------------------------------------------------------------------------

void AppearEarth (int x, int y, int sizeX, int sizeY, COLORREF from, COLORREF to,
                  int time, int steps)
      {
      int   r0 = txExtractColor (from, TX_RED),   r1 = txExtractColor (to, TX_RED),
            g0 = txExtractColor (from, TX_GREEN), g1 = txExtractColor (to, TX_GREEN),
            b0 = txExtractColor (from, TX_BLUE),  b1 = txExtractColor (to, TX_BLUE);

      for   (int i = 0; i <= steps && !_kbhit(); i++)
            {
             int  r = r0 + (r1 - r0) * i/steps,
                  g = g0 + (g1 - g0) * i/steps,
                  b = b0 + (b1 - b0) * i/steps;

            DrawEarth (x, y, sizeX, sizeY, RGB (r, g, b));

            Sleep (time / steps);
            }
```

```
      kbget();

      DrawEarth (x, y, sizeX, sizeY, to);
      }

//---------------------------------------------------------------------------

void AppearText    (int x, int y, const char* text, COLORREF from, COLORREF to,
                    int time, int steps)
      {
      int   r0 = txExtractColor (from, TX_RED),   r1 = txExtractColor (to, TX_RED),
            g0 = txExtractColor (from, TX_GREEN), g1 = txExtractColor (to, TX_GREEN),
            b0 = txExtractColor (from, TX_BLUE),  b1 = txExtractColor (to, TX_BLUE);

      for   (int i = 0; i <= steps && !_kbhit(); i++)
            {
            int   r = r0 + (r1 - r0) * i/steps,
                  g = g0 + (g1 - g0) * i/steps,
                  b = b0 + (b1 - b0) * i/steps;

            txSetColor (RGB (r, g, b));
            txTextOut (x, y, text);

            Sleep (time / steps);
            }
```

```
      kbget();

      txSetColor (to);
      txTextOut (x, y, text);
      }

//---------------------------------------------------------------------------

void DrawFlag (int x, int y, int sizeX, int sizeY, COLORREF color, COLORREF bkColor)
      {
      txSetColor (color);
      txSetFillColor (bkColor);

      txLine (x, y, x, y - sizeY);
      txRectangle (x, y - sizeY/2, x + sizeX, y - sizeY);

      txSelectFont ("Times New Roman", sizeX/2 + 1);
      txTextOut (x + sizeX/2, y - sizeY*7/8, "C++");
      }

//---------------------------------------------------------------------------

void DrawHello (int x, int y, const char* text, int size, COLORREF color)
      {
      txSetColor (color);
```

```
	txSelectFont ("Times New Roman", size);
	txSetTextAlign (TA_CENTER);

	txTextOut (x, y, text);
	}

//---------------------------------------------------------------------------

void DrawFrame (int sizeX, int sizeY, int size, COLORREF color)
	{
	txSetColor (color);
	txSetFillColor (TX_TRANSPARENT);

	txRectangle (size, size, sizeX-size, sizeY-size);
	}

//---------------------------------------------------------------------------

int kbget()
	{
	int ch = 0;
	while (_kbhit()) ch = _getch();
	return ch;
	}
```