Федеральное государственное автономное
образовательное учреждение
высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»
Институт космических и информационных технологий
институт
Вычислительная техника
кафедра

УТВЕРЖДАЮ
Заведующий кафедрой

_____ А.И. Легалов
подпись инициалы, фамилия
«____» июня 2016 г.

БАКАЛАВРСКАЯ РАБОТА

09.03.01 «Информатика и вычислительная техника»
код и наименование направления

Разработка кроссплатформенной графической библиотеки для начинающих программистов
тема

Пояснительная записка

Научный руководитель	_____	д.т.н., профессор	А.И. Легалов
	подпись, дата	должность, ученая степень	инициалы, фамилия
Выпускник	_____		Е.А. Гаевский
	подпись, дата		инициалы, фамилия
Нормоконтролер	_____		В.И. Иванов
	подпись, дата		инициалы, фамилия

Красноярск 2016

АННОТАЦИЯ

Целью выпускной квалификационной работы является разработка учебной библиотеки функций, обеспечивающей создание простых графических приложений на первоначальном этапе обучения. Для совместимости интерфейс разрабатываемой библиотеки совпадает с интерфейсом библиотеки функций TXLib, что позволяет использовать уже существующие методические и программные наработки. Для переносимости на различные операционные системы и компьютерные архитектуры учебная библиотека реализована с использованием библиотеки Qt и, по сути, выступает в качестве дополнительной программной обертки.

Выпускная квалификационная работа изложена на 54 страницах основного текста и состоит из введения, четырех разделов, заключения, списка использованных источников, содержит 30 рисунков, 6 таблиц, 5 приложений. Прилагаемый CD-диск содержит презентацию, исходные тексты разработанной библиотеки и демонстрационных примеров.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Одной из важных задач при обучении программированию является обеспечение наглядности отображения результатов работы программ, их удобное взаимодействие с конечным пользователем. Графические элементы оформления программ делают процесс обучения более наглядным и интересным. На сегодняшний день существует целый ряд средств графического оформления, однако современные графические библиотеки, как правило, достаточно громоздки, сложны для новичков, имеют высокий порог вхождения, что не позволяет использовать их на начальном этапе обучения программированию [1]. Очень часто это мешает начинающим программистам сконцентрироваться на решении основной задачи, вынуждает их тратить дополнительное время и силы на ее графическую визуализацию.

Среди существующего многообразия средств графического оформления особого внимания заслуживает библиотека «The Dumb Artist Library» (TX Library, TXLib), ориентированная, прежде всего, на обучение программированию и на новичков. Она была создана с целью помочь начинающим в изучении простейших принципов программирования. Философия TX Library - облегчить первые шаги в программировании и подтолкнуть к творчеству и самостоятельности [1]. Она проста в освоении, компактна, работать с ней легко и удобно даже непрофессионалам. При всех достоинствах TXLib, у нее все же есть недостаток, который в современном мире разнообразия устройств, платформ и операционных систем становится все более существенным: реализация TXLib такова, что библиотека работает только под управлением операционной системы Microsoft Windows. Таким образом, при всей массе достоинств ее невозможно использовать на других операционных системах. Вместе с тем, по мере того как различные портативные и мобильные устройства внедряются в образовательный процесс, а свободные операционные системы находят широкое применение, переносимость библиотек с одной операционной системы на другую и их

платформенная независимость играет все более существенную роль. Поэтому актуальной является задача создания графической библиотеки, обеспечивающей первоначальное обучение программированию на разнообразных аппаратно-программных платформах.

В связи с вышесказанным для организации эффективного обучения на различных платформах необходима графическая библиотека, сочетающая в себе, с одной стороны, кроссплатформенность, гибкость и разнообразие возможностей, а с другой, легкость, простоту понимания и использования.

Использование начинающими программистами этой библиотеки в своих программах освободило бы их от преждевременных ненужных сложностей, ускорило разработку их проектов, а также послужило важным подготовительным этапом на пути ознакомления с более сложными средствами разработки графических приложений.

Целью выпускной квалификационной работы является разработка переносимой графической библиотеки, сочетающей в себе разнообразие возможностей по созданию простых графических приложений и простоту использования, что позволило бы применять ее на начальных этапах обучения программированию.

Для достижения поставленной цели в работе решались следующие задачи:

- на основе изучения и анализа существующих графических библиотек проведен выбор средств, используемых для реализации проекта;
- проведено согласование интерфейса библиотеки TXLib с интерфейсом библиотеки Qt, выбранной для разработки в ходе проведенного анализа;
- разработана кроссплатформенная графическая библиотека QTXLib, реализующая основные функции библиотеки TXLib;
- рассмотрены различные методы подключения разработанной библиотеки к программе пользователя;
- проведено тестирование совместимости исходной и разработанной библиотек на уже существующих примерах;
- разработан ряд собственных примеров использования библиотеки.

Результатом выполнения выпускной квалификационной работы является переносимая библиотека QTXLib, написанная на базе кроссплатформенного инструментария Qt, подключение к программе пользователя которой может осуществляться с применением различных режимов.

Первый раздел выпускной квалификационной работы раскрывает общее описание графической библиотеки TXLib и ее функциональные возможности,

Второй раздел посвящен краткому описанию кроссплатформенного инструментария Qt на основе которого реализуется проект.

В третьем разделе рассматривается непосредственное построение библиотеки QTXLib, раскрывается ее внутренняя архитектура и особенности реализации, а также использование и компоновка с программами конечного пользователя.

В четвертом разделе рассматривается применение библиотеки для разработки прикладных программ на конкретных примерах, а также рассматриваются перспективы дальнейшего развития библиотеки QTXLib.

1 Особенности библиотеки TXLib

Подробное описание библиотеки TX Library представлено в [1]. Ниже приводятся основные ее характеристики, взятые из этого источника.

TX Library - компактная графическая библиотека, написанная для Win32 на C++. Это инструмент, небольшая "песочница" для начинающих, реализована с целью помочь им в изучении простейших принципов программирования. Философия TX Library - облегчить первые шаги в программировании и подтолкнуть к творчеству и самостоятельности.

Принципы, заложенные в TXLib для повышения качества обучения [1]:

- сделай сам. В TXLib многие вещи сделаны или оставлены не совсем удобными для применения. Это – предложение подумать, как сделать это самому, и, как правило, для этого в TXLib есть средства;
- загляни в Help;
- посмотри, как сделано. Загляни в код. Он создавался в том числе как пример программной системы со своей логикой и со своей реализацией, а некоторые функции можно понять только по коду, потому что их нет в системе помощи;
- посмотри, как сделано иначе. TXLib – не единственная графическая библиотека, и реализация "простого графического холста", примененная в ней – не единственное решение. Посмотри, как устроены десятки других графических библиотек;
- выйди за пределы "песочницы". Это усиление принципа "сделай сам".

Библиотека TXLib представляет собой заголовочный файл TXLib.h, который содержит определения перечислений, макросов, функций, а также нескольких классов. Файл библиотеки подключается к программе пользователя с помощью директивы препроцессора #include.

Все функции библиотеки условно можно разделить на группы:

- функции для рисования;
- функции для работы с мышью;

- функции, реализующие диалоговые окна;
- разные прочие функции;
- служебные функции;
- функции для получения технической информации.

Классы библиотеки TXLib:

- txDialog - базовый класс для диалоговых окон;
- txDialog:Layout — класс для описания элемента диалогового окна;
- txAutoLock — класс для автоматической блокировки и разблокировки критической секции.

Функции, константы и макросы библиотеки TXLib, а также их краткое описание приведены в приложении А.

2 Обзор возможных средств переносимой реализации TXLib и его выбор

2.1 Возможные библиотеки, обеспечивающие реализацию интерфейса TXLib

Для реализации проекта доступен целый ряд средств графического оформления программ, наиболее заметными из которых являются:

- Qt [27];
- Juce [3];
- FOX Toolkit [4];
- FLTK [5];
- wxWidgets [6];
- GTK+ [7].

2.1.1 Библиотека Qt Library

Qt — один из крупнейших на сегодняшний день, признанных лидеров среди инструментариев для создания современных графических приложений. В состав Qt, помимо графических библиотек, входит еще масса полезных инструментов, таких как поддержка работы со звуком, сетью, потоками, базами данных и многие другие, охватывающих практически все потребности разработчиков. Инструмент кроссплатформенный, написан на языке программирования C++, имеет «привязки» ко многим другим языкам программирования: Python - PyQt, Ruby - QtRuby, Java - Qt Jambi, PHP - PHP-Qt и другим [2, 27]. Qt имеет огромный массив подробнейшей документации, в том числе на русском языке, что, безусловно, является его достоинством. Надежен, проверен временем, активно развивается, периодически выходят новые версии. Также к достоинствам инструментария можно отнести его широкое распространение. С использованием Qt написаны такие популярные

программы как Skype [28], VirtualBox [29], среда рабочего стола KDE [30] и многие другие известные приложения.

Qt gозволяет запускать написанное с его помощью ПО в большинстве современных операционных систем путём простой компиляции программы для каждой ОС без изменения исходного кода. Включает в себя все основные классы, которые могут потребоваться при разработке прикладного программного обеспечения, начиная от элементов графического интерфейса и заканчивая классами для работы с сетью, базами данных и XML. Qt является полностью объектно-ориентированным, легко расширяемым и поддерживающим технику компонентного программирования [20].

Существуют версии библиотеки для Microsoft Windows, систем класса UNIX с графической подсистемой X11, Mac OS X, Microsoft Windows CE, встраиваемых Linux-систем и платформы S60 [21].

Отличительная особенность Qt от других библиотек — использование Meta Object Compiler (MOC) — предварительной системы обработки исходного кода [31] (в общем-то, Qt — это библиотека не для чистого C++, а для его особого наречия, с которого и «переводит» MOC для последующей компиляции любым стандартным C++ компилятором). MOC позволяет во много раз увеличить мощь библиотек, вводя такие понятия, как слоты и сигналы. Кроме того, это позволяет сделать код более лаконичным. Утилита MOC ищет в заголовочных файлах на C++ описания классов, содержащие макрос Q_OBJECT, и создаёт дополнительный исходный файл на C++, содержащий метаобъектный код [9].

Qt – взаимосвязанная система. Родственность Qt-объектов осуществляется через наследование класса QObject, а связи между ними через сигнально-слотовую систему. Управление же объектами осуществляется с помощью событий [9].

В библиотеке Qt используется терминология виджета (от англ. widget — штуковина, приспособление), как элемента GUI-интерфейса. Каждый отдельный элемент управления интерфейса на экране является виджетом.

Каждый виджет может настраиваться посредством изменения его свойств. С помощью свойств можно указать размеры виджетов, их расположение, особенности внешнего вида и др.

К недостаткам Qt можно отнести большой объем библиотеки, ее «многогранность», порой излишняя функциональность, нагроможденность, и, как следствие, сложность освоения и использования начинающими программистами. На рисунке 2.1 представлена схема модулей библиотеки. Более подробно с описанием инструментария Qt можно ознакомиться на домашней странице проекта [27].

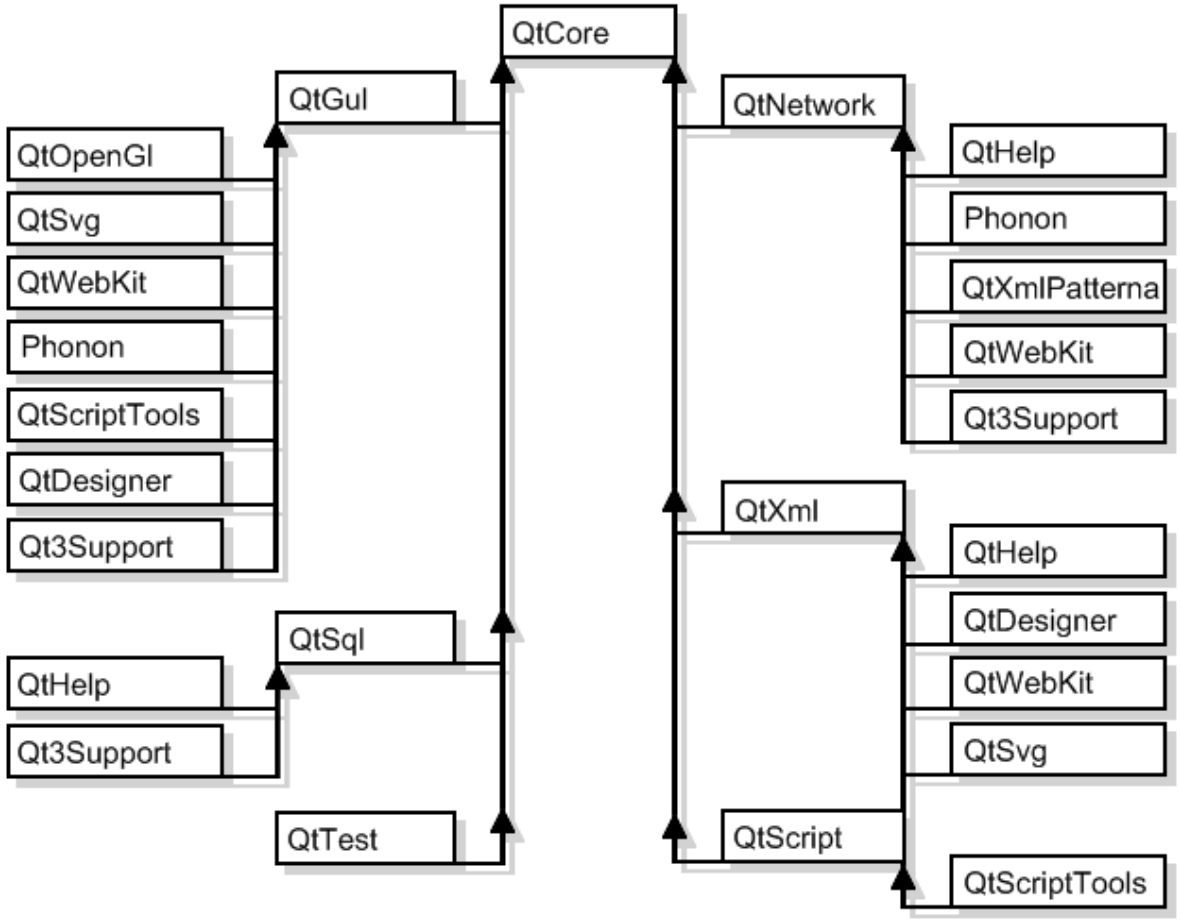

Рисунок 2.1 — Схема модулей Qt

2.1.2 Библиотека Juce

Juce — еще один кроссплатформенный инструментарий, используемый для разработки приложений с графическим интерфейсом [11].

С официального сайта разработчиков [3]: «JUCE (Jules' Utility Class Extensions) - это всеохватывающая библиотека классов C++ для разработки кроссплатформенного программного обеспечения. Он содержит практически все что вам может понадобиться для создания большинства приложений, особенно хорошо подходит для построения сложных GUI, обработки графики и аудио». Программы написанные с помощью последних версий Juce одинаково работают на Windows, Mac Os и Linux, на платформах iPhone и Android [3].

Также как и Qt, содержит классы позволяющие программе работать с графикой и звуком, разбирать XML, работать с сетью и криптографией и т. д. За счёт этого нуждающиеся в дополнительных библиотеках программисты могут использовать только библиотеку Juce, или хотя бы сократить количество используемых ими сторонних библиотек. Содержит большой набор функций для работы со звуком. OpenSource лицензия Juce появилась в 2003 году, а с 2005 фреймворк обзавелся ее платной версией для закрытого ПО.

Стоит заметить что Juce представляет собой не набор библиотек в привычном смысле или набор исходных текстов из которых вы должны собрать библиотеки, а набор заголовочных файлов и файлов исходного кода, содержащие все предоставляемые фреймворком классы. Данный подход не только упрощает миграцию с одной платформы на другую, избавляя вас от необходимости включать в состав дистрибутивов библиотеки фреймворка или требовать их наличия на компьютере пользователя, но и решает проблему разрядности библиотек от которых зависит фреймворк [14].

Недостатками библиотеки являются сложность освоения, малое количество практических примеров даже в документации.

2.1.3 Библиотека Fox Toolkit

FoxToolkit — кроссплатформенная библиотека классов для построения графического интерфейса пользователя. Один из самых быстрых пакетов, содержит большое число элементов GUI и поддержку OpenGL, в том числе

трехмерной графики [12]. Библиотека ориентирована исключительно на работу с графикой, а по сложности использования близка к Qt, что можно отнести к ее недостаткам. Имеет неплохую документацию. Более подробно с библиотекой FoxToolkit можно ознакомиться на домашней странице проекта [4].

2.1.4 Библиотека FLTK

FLTK — аббревиатура Fast Light Toolkit — быстрый легковесный инструментарий — кроссплатформенная библиотека инструментов для построения графического интерфейса пользователя [13]. Библиотека использует свои собственные независимые системы виджетов что позволяет писать программы одинаково выглядящие и работающие на разных операционных системах [5]. В отличие от других подобных библиотек FLTK ограничивается только графической функциональностью, поэтому имеет малый размер. Не использует сложных макросов, препроцессоров, шаблонов и пр., что вкупе с малым размером кода, облегчает использование библиотеки не очень искушенными пользователями. Продолжением достоинств являются недостатки библиотеки, такие как меньшее число виджетов, несколько упрощенная графика и невозможность сборки приложения, выглядящего естественно под конкретной операционной системой. Б. Страуструп (B. Stroustroup) в своей книге [30] приводит примеры обучения программированию с использованием именно этой библиотеки. Кроме того, дополнительный полезный материал можно найти на его домашней странице [32].

На рисунке 2.2 приведена схема организации кода с использованием графической библиотеки FLTK [15].

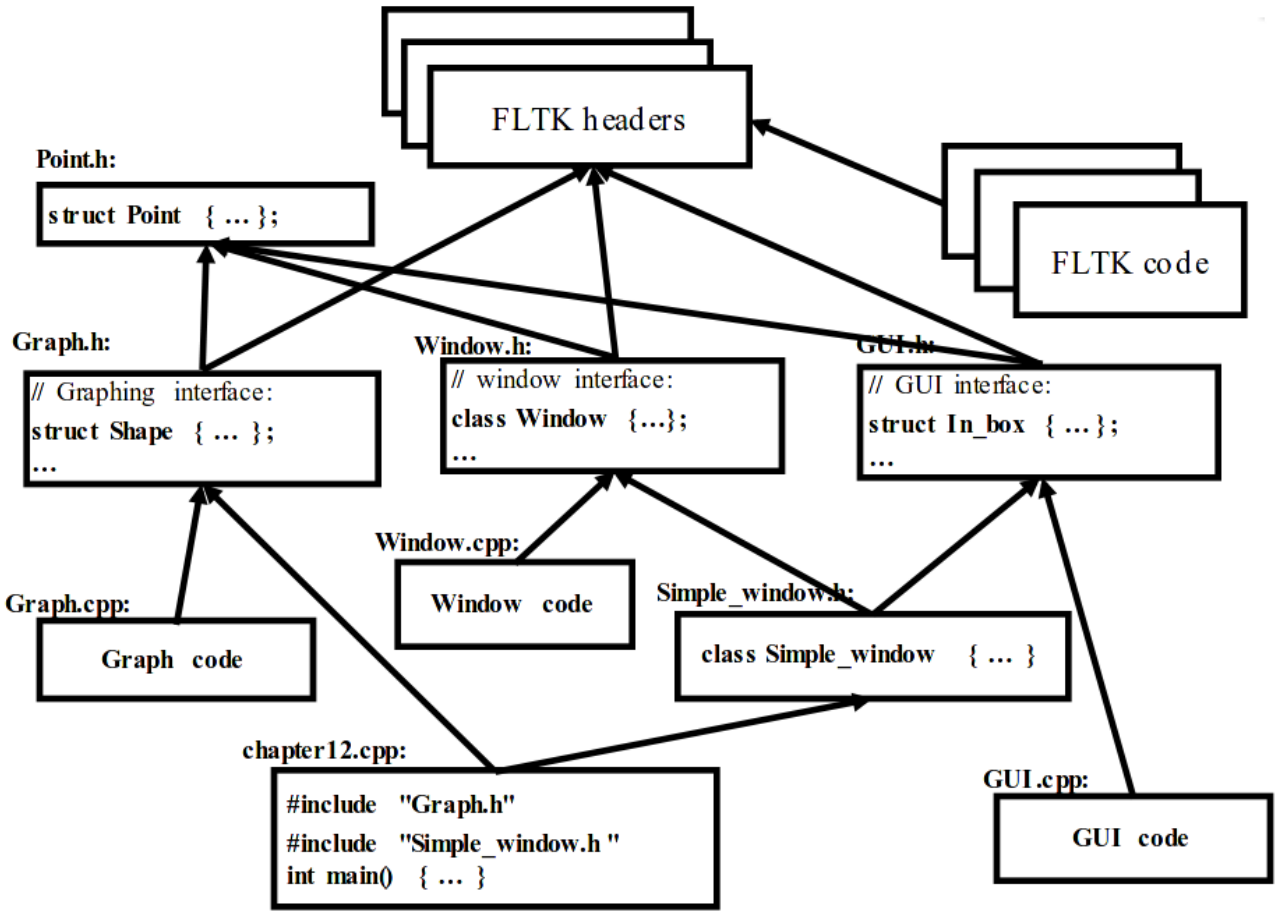

Рисунок 2.2 — Схема организации кода с использованием FLTK

Более подробно с описанием графической библиотеки можно ознакомиться на официальном сайте проекта [5].

2.1.5 Библиотека wxWidgets

wxWidgets — кроссплатформенная библиотека инструментов для построения графического интерфейса пользователя. Помимо графического оформления приложений включает в себя и множество других функций, таких поддержка XML и HTML, работа с архивами, файловыми системами, процессами, подсистемами печати, мультимедиа, сетями, и другие, что позволяет создавать на ее основе весьма разнообразное программное обеспечение. В отличие от некоторых других библиотек максимально использует «родные» графические элементы интерфейса операционной системы всюду, где это возможно так, чтобы приложения под различными

операционными системами выглядели одинаково. Получила широкую популярность благодаря своей бесплатности даже для коммерческого использования. Как и все библиотеки широкой функциональности достаточно сложна в освоении и использовании, что можно отнести к недостаткам. С использованием wxWidgets написаны такие популярные приложения как FileZilla [33], Audacity [34], BitTorrent [35] и многие другие. По библиотеке доступна исчерпывающая документация [16]. Более подробно с описанием библиотеки можно ознакомиться на сайте разработчиков [6].

2.1.6 Библиотека GTK+

GTK+ [7] - объектно-ориентированный, кроссплатформенный инструментарий для создания графического интерфейса пользователя. Изначально создавался для нужд графического редактора GIMP (GNU Image Manipulation Program), но потом отделился в самостоятельный проект. Состоит из двух компонентов: GTK (GIMP Toolkit, GIMP library), содержащей непосредственно набор элементов графического интерфеса — виджетов, и GDK (GIMP Drawing Toolkit) который отвечает за вывод виджетов на экран и может использовать для этого функции X Window System, WinAPI или Mac OS X. Инструментарий имеет давнюю историю и продолжает активно развиваться. Отдельно следует отметить хорошую поддержку интернационализации и локализации. К недостаткам можно отнести сложности с установкой библиотек и языковых привязок под Windows. Архитектура GTK+ представлена на рисунке 1.3.

Рисунок 1.3 — Архитектура GTK+

Структура GTK+:

- GTK+ - иерархие виджетов;
- GDK — кроссплатформенные средства отрисовки и управления окнами;
- GLIB — обеспечивает структуры данных, макросы, утилиты, абстракцию основного цикла и т. п.;
- Pango — используется для интернационализации текстов;
- ATK — реализует средства для людей с ограниченными возможностями;
- GIO — реализует VFS API.

С использованием GTK+ написана GNOME - оконная среда Linux-систем [36], а также приложения работающие в этой среде. Имеет обширную документацию [7].

2.2 Выбор библиотеки для реализации интерфейса TXLib

Среди всех вышеперечисленных средств разработки графических приложений для реализации данного проекта был выбран инструментарий Qt как наиболее популярный и распространенный, широко используемый в различных сферах, надежный, с широким набором возможностей, использующий современные технологии в области программирования, хорошо зарекомендовавший себя в сфере разработки современного программного обеспечения. Выбирая Qt в качестве инструмента для разработки библиотеки

QTXLib, мы автоматически унаследуем необходимую нам переносимость - кроссплатформенность, присущую Qt. Стоит отметить что Qt имеет хорошую документацию на русском языке, а также хорошую поддержку в виде многочисленных форумов и сообществ, в том числе русскоязычных [8]. Qt сегодня — это целое сообщество, объединяющее разработчиков и пользователей, работающих вместе на благо развития современного кроссплатформенного программного обеспечения. А учитывая текущую тенденцию проводимого учебного процесса и работу ряда научных групп кафедры ВТ, разрабатывающих графические и консольные приложения с использованием именно этой библиотеки, выбор Qt оправдан возможностью последующего развития проекта и использования QTXLib на кафедре. Кроме того, такой выбор значительно облегчит возможность дальнейшего перехода к обучению библиотеке Qt студентов.

2.3 Средства библиотеки Qt, используемых для реализации QTXLib

Из всего разнообразиия возможностей Qt, для реализации графической библиотеки QTXLib будем использовать его систему рисования и виджеты для построения диалоговых окон.

Система рисования Qt позволяет рисовать на экране и печатающих устройствах используя один и тот же API, и основана, в основном, на классах QPainter, QPaintDevice и QpaintEngine. QPainter используется для выполнения операций рисования, QPaintDevice - абстракция двухмерного пространства, на котором можно рисовать используя QPainter, а QPaintEngine предоставляет интерфейс, который рисовальщик использует для рисования на различных типах устройств. Класс QPaintEngine используется для внутренних нужд в QPainter и QPaintDevice, и невидим для программистов приложений если только они не создадут свой собственный тип устройства [10].

QPainter содержит функции для выполнения большей части рисования, необходимого программам с графическим пользовательским интерфейсом. Он

может нарисовать всё, начиная с простых графических примитивов (представляемых классами QPoint, QLine, QRect, QRegion и QPolygon) и заканчивая сложными фигурами, например, векторными траекториями. В Qt векторные траектории представлены классом QPainterPath. QPainterPath предоставляет контейнер для операций рисования, позволяющий создавать и повторно использовать графические фигуры [10].

Линии и контуры рисуются используя класс QPen. Кистью пера является объект QBrush, используемый для заполнения штрихов, создаваемых пером, т.е. класс QBrush определяет узор-заполнитель. QPainter также может рисовать выровненный текст и растровые изображения.

В качестве альтернативы рисования, Qt предоставляет модуль QtOpenGL, предлагающий классы, которые делают более легким использование OpenGL в приложениях Qt. Помимо прочего, модуль предоставляет класс виджета OpenGL, который может использоваться также как и любой другой виджет Qt, за исключением того, что он открывает дисплейный буфер OpenGL, где может быть использовано OpenGL API для формирования изображения [10].

Все виджеты в Qt содержат палитру и используют свою палитру для отрисовки самих себя. Палитра виджета представляется классом QPalette, который содержит группы цветов для каждого состояния виджета. Все цвета в Qt представляются классом QColor, который поддерживает цветовые модели RGB, HSV и CMYK.

При отрисовке текста шрифт задается используя класс QFont. Атрибуты шрифта, которые действительно используются, можно получить используя класс QFontInfo. Кроме того, класс QFontMetrics предоставляет размеры шрифта, а класс QFontDatabase предоставляет информацию о шрифтах, доступных в основной оконной системе [10].

Классы Qt, используемые при реализации QTXLib приведены в приложении Б.

3 Разработка графической библиотеки QTXLib

3.1 Программный интерфейс библиотеки и связь с Qt

Графическая библиотека QTXLib представляет собой программу на языке C++. Библиотека использует средства объектно-ориентированного программирования и реализована виде нескольких заголовочных файлов, описание которых приведено в таблице 3.1. Все файлы библиотеки для удобства помещены в один каталог с именем qtx.

Таблица 3.1 — Состав библиотеки QTXLib

№ п/п	Имя файла	Описание файла
1	libqtx.h	Основной заголовочный файл, именно он включается в программу пользователя. В зависимости от пользовательского определения EXTERN_QTX включает или не включает в проект файлы исходного кода библиотеки QTXLib.
2	qtxextern.h	Содержит определения внешних публичных функций библиотеки QTXLib. Используется если определен EXTERN_QTX, т. е. сборка программы пользователя производится с использованием динамической библиотеки libqtx.so, без использования файлов исходного кода библиотеки QTXLib, в противном случае используется qtxwrapper.h.
3	qtxwrapper.h	Файл содержит исходный код публичных функций пользователя, является оберткой над классом Qtx. Используется, если пользователем не определено EXTERN_QTX, т.е сборка программы производится без использования динамической библиотеки libqtx.so, в противном случае используется qtxextern.h.
4	txdefs.h	Содержит глобальные определения констант, макросов, перечислений, переменных из библиотеки txLibrary.
5	qtxglobal.h	Содержит включение заголовочных файлов библиотеки Qt, используемой для реалиазации проекта.
6	qtxpoint.h	Содержит определение класса точки QTxPoint.
7	qtxpoint.cpp	Содержит реализацию методов для класса точки QTxPoint.
8	qtx.h	Содержит определение класса Qtx, в котором реализован графический функционал библиотеки и функции взаимодействия с пользователем.
9	qtx.cpp	Содержит реализацию методов класса Qtx.

Окончание таблицы 3.1

№ п/п	Имя файла	Описание файла
10	qtxcnt.h	Содержит определение класса QTxContainer, в котором реализован диалогово-событийный функционал библиотеки.
11	qtxcnt.cpp	Содержит реализацию методов класса QTxContainer.
12	examples.zip	Архив примеров программ, использующих библиотеку QTXLib.
13	build	Исполняемый bash-скрипт для автоматической сборки и компиляции примера из исходников.
14	build_ext	Исполняемый bash-скрипт для автоматической сборки и компиляции примера с использованием динамической библиотеки libqtx.so, без использования файлов исходного кода.
15	libqtx.so	Файл динамической библиотеки QTXLib, подключаемый к программам пользователя с помощью опции -l компилятора gcc.

Использование библиотеки QTXLib начинается с включения в программу пользователя директивой препроцессора #include файла qtx/libqtx.h. При этом, если компилировать и собирать программу планируется с использованием динамической библиотеки libqtx.so, т. е. без включения исходного кода библиотеки, то перед включением файла libqtx.h необходимо определить QTX_EXTERN с помощью директивы препроцессора #define. Зависимость заголовочных файлов библиотеки QTXLib показана на рисунке 3.1.

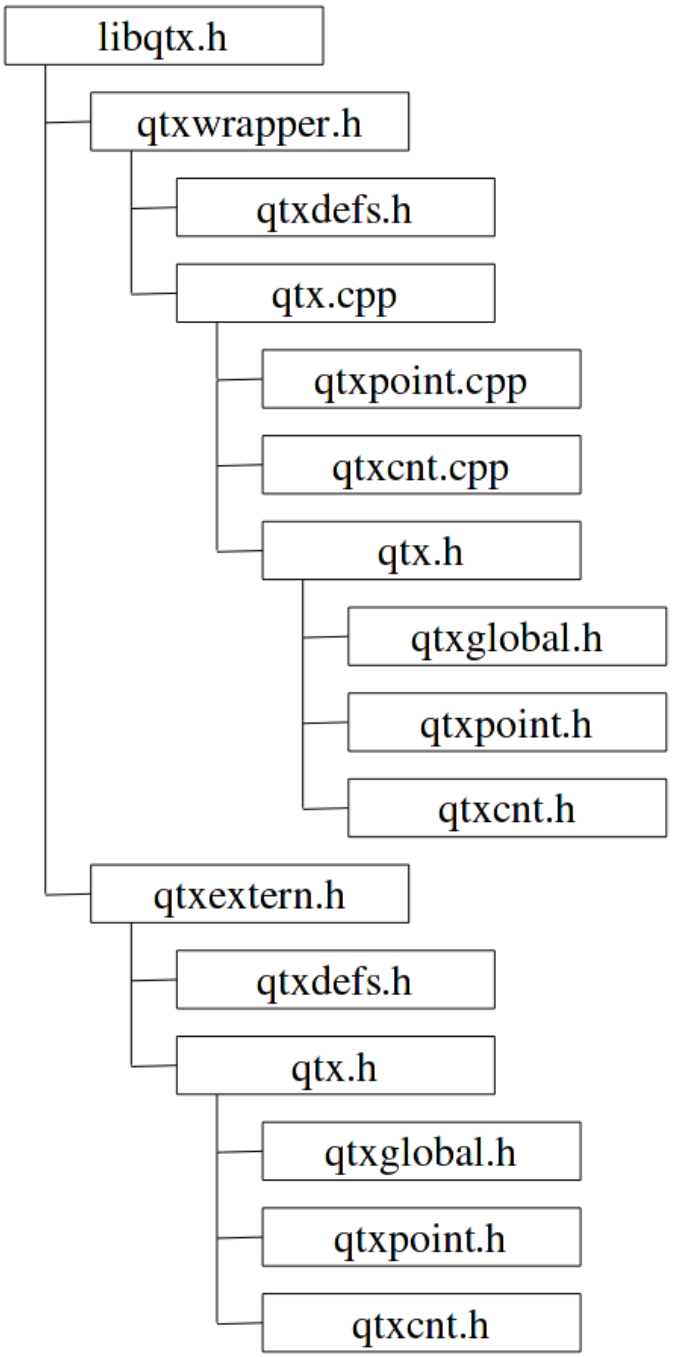

Рисунок 3.1 — Зависимость исходных файлов библиотеки QTXLib

Весь внешний (пользовательский) программный интерфейс библиотеки QTXLib описан в файлах qtxwrapper.h (с включением исходного кода) и qtxextern.h (без включения исходного кода, содержит только прототипы функций для внешних вызовов).

Для связи с классами Qt библиотека использует соответствующие

заголовочные файлы инструментария. Более наглядно это отображено в таблице 3.2.

Таблица 3.2 — Связь библиотеки QTXLib с Qt

Файл QTXLib	Заголовочные файлы Qt
qtxglobal.h	QtCore/qglobal.h QApplication QPainter QImage QDesktopWidget QGraphicsTextItem QTime QStack QMessageBox QInputDialog QSystemTrayIcon QBuffer
qtxpoint.h	QPoint
qtxcnt.h	QLabel QKeyEvent QMouseEvent QPushButton QHBoxLayout QVBoxLayout QTextEdit QLineEdit QCheckBox QRadioButton QButtonGroup QGroupBox QComboBox QListWidget

Кроме всего прочего, необходимо также отметить, что некоторые функции QTXLib, в частности функции создания и размещения в диалоговом окне элементов управления диалога (кнопки, строки ввода и т. п.), возвращают указатели на объекты классов Qt. Это дает пользователю доступ ко всему набору свойств и методов таких объектов, доступных в Qt, а вместе с этим и невероятную гибкость настройки интерфейса, ограниченную лишь возможностями самого инструментария Qt,

3.2 Особенности реализации

При всем удобстве и простоте библиотеки TXLibrary, аналогом которой является QTXLib, эта библиотека все же имеет и ряд существенных недостатков, обойти стороной и не обратить внимания на которые было бы неправильно. Все эти недостатки и недочеты описаны ниже и учтены при реализации QTXLib. Справедливости ради нужно заметить, что таких особенностей (несовместимостей с TXLibrary) совсем немного. Итак, разберем все эти особенности реализации QTXLib на конкретных наглядных примерах.

3.2.1 Завершение главной функции

Функция main()

Как она выглядит в txLib:

```
int main()
{
	txCreateWindow(800, 600);
	txSetColor(TX_GREEN);
	txLine(0,0,800,600);
	return 0;
}
```

А вот так этот же код выглядит в QTXLib:

```
int main()
{
	txCreateWindow(800, 600);
	txSetColor(TX_GREEN);
	txLine(0,0,800,600);
	return txExec();
}
```

Отличие здесь лишь наличии в последнем примере функции txExec() которая запускает главный рабочий цикл Qt обработки событий (App → exec()).

Кроме того, в txCreateWindow() добавлен необязательный параметр

который позволяет установить пользовательский заголовок окна. Необходимо заметить, что библиотека TXLib по умолчанию создает окно черного цвета (TX_BLACK), тогда как при использовании библиотеки QTXLib окно всегда создается цветом, взятым из системной палитры цветов и соответствующему цвету диалоговых окон в системе.

В практике программирования может возникнуть случай, когда какие-либо объекты классов Qt могут потребоваться пользователю до создания главного окна. В этом случае необходимо вызвать функцию txInit() - она не создает окон, но создаст динамический объект класса qtx и инициализирует QApplication с заданным заголовком окна приложения, что позволить использовать функционал Qt до создания самого окна.

3.2.2 Работа с несколькими окнами

Следующая полезная особенность QTXLib — возможность работы одновременно с несколькими окнами. Библиотека TXLibrary такой возможности, к сожалению, не имеет и ограничивает пользователя одним единственным рабочим окном, что, конечено же, является существенным недостатком при разработке современного программного обеспечения. Для решения этой проблемы в QTXLib добавлены динамический список созданных окон, указатель на активное окно, функция txSetWindow() которая позволяет сделать окно активным, а функция txCreateWindow() всегда возвращает указатель на созданное окно. Вот пример использования нескольких окон в программе:

```
TXWINDOW w1 = txCreateWindow(800, 600, «1st window»);
TXWINDOW w2 = txCreateWindow(200, 200, «2nd window»);
txSetWindow(w1);          // устанавливает активным первое окно
txLine(0, 0, 100, 100);
txSetWindow(w2);          // устанавливает активным второе окно
txLine(0, 0, 100, 100);
```

Таким образом, при использовании QTXLib пользователь не ограничен

работой с одним окном, может создавать их столько, сколько ему необходимо для решения конкретной задачи. Список создаваемых окон динамический и ограничен только объемом свободной памяти компьютера.

3.2.3 Обработка событий

Еще одной важной особенностью QTXLib является иная, нежели чем у TXLibrary организация обработки событий. В TXLibrary применяется обычный пользовательский цикл обработки, в котором вызываются функции WinAPI для опроса состояния мыши, клавиатуры, а задержка по таймеру реализуется функцией Sleep(). Например:

```
int main()
{
	txCreateWindow(0, 0, 800, 600);
	while(!GetAsyncKeyState(VK_ESCAPE))
	{
		if(GetAsyncKeyState(VK_DOWN))
		{
			// doing something
		}
		if(GetAsyncKeyState(VK_UP))
		{
			// doing something else
		}
		// mouse events handling
		printf(«Current mouse state is: [%d, %d, %d]\n»,
				xMouseX(), txMouseY(), txMouseButtons());
		Sleep(100); // timer delay emulation
	}
	return 0;
}
```

При работе с современными операционными системами такая обработка событий не исключена и возможна, но не является хорошим решением, т. к.

кроме событий пользователя в системе одновременно могут возникать и многие другие события, требующие своевременной обработки, такие как, например, события отрисовки окон и элементов интерфейса, а также различные события самой операционной системы. Поэтому в QTXLib основной рабочий цикл программы специально организоваывать не нужно - он уже существует в самой операционной системе и в Qt, и задача программиста — лишь добавить в этот цикл обработчики своих собственных событий, не нарушая работу системы.

Вот так обработка событий выглядит в QTXLib:

```
void keyHandler(int key)
 {
      if(key == Qt::Key_Up)
      {
           // doing something
      }
      if(key == Qt::Key_Down)
      {
           // doing something else
      }
 }
void mouseHandler(int x, int y, int button, bool dbclick)
 {
      printf(«Curr. mouse state is: [%d, %d, %d]\n»,
                                          x, y, button);
 }
void timerHandler(void *p)
 {
      // doing something timer relative (every 100 ms)
 }
int main()
 {
      txCreateWindow(800, 600);
      txKeyEvent(keyHandler);
      txMouseEvent(mouseHandler);
      txTimerEvent(timerHandler, NULL, 100);
```

```
    return txExec();
}
```

Таким образом, мы видим, что для обработки событий клавиатуры, мыши и таймера достаточно лишь сделать своеобразные «подписки» - установить свои функции-обработчики этих событий, и при их наступлении управление будет передано этим обработчикам.

txKeyEvent(...) подключает обработчик клавиатуры, в нашем случае keyHandler().

txMouseEvent(...) подключает обработчик мыши, в нашем случае mouseHandler().

txTimerEvent(...) подключает обработчик события по таймеру, в нашем случае timerHandler() с задержкой 100 мс., его можно использовать для реализации любых параллельных/фоновых процессов в программе происходящих по таймеру, например в играх. При этом в обработчик можно передавать указатель на какую-нибудь структуру параметров, что тоже очень полезно.

txExec() запускает главный рабочий цикл Qt обработки событий.

3.2.4 Построение диалоговых окон

Следующая важная и интересная особенность библиотеки QTXLib — иная, нежели в TXLibrary, более простая, наглядная и удобная для новичков работа с диалоговыми окнами и их элементами. Для того чтобы пользователю создать диалоговое окно с помощью библиотеки TXLib, ему необходимо определить класс, наследующий от класса txDialog, а в нем переопределить некоторые методы. Это заведомо подразумевает наличие у пользователя базовых знаний по объектно-ориентированному программированию, которое, скорее всего, отсутствует у начинающих. По этой причине создание диалоговых окон так, как предлагает TXLibrary, через наследование классов, может вызвать у пользователя некоторые затруднения. Кардинально иной способ предлагает

библиотека QTXLib. Окно, созданное функцией txCreateWindow() уже может являться контейнером для элементов диалога. Таким образом у пользователя появляется полезная возможность совмещать в едином пространстве как графику, так и элементы управления диалогового окна. Сами элементы диалога, различные кнопки, строки ввода, метки, группы и прочее, добавляются путем простого вызова соответствующих функций библиотеки. Кроме того, для удобства пользователя перечень элементов управления диалога, доступных для размещения в диалоговом окне в QTXLib расширен, по сравнению с TXLibrary.

Элементы диалога, доступные в TXLibrary:

- BUTTON — кнопка;
- EDIT — редактируемый текст (строка ввода);
- STATIC — нередактируемый элемент (картинка, текст и т. д.)
- SCROLLBAR — полоса прокрутки;
- LISTBOX — список с прокруткой;
- COMBOBOX — комбинированный список.

А это элементы диалога, доступные в библиотеке QTXLib:

- Button — кнопка;
- TextEdit — редактор многострочного текста с прокруткой;
- Edit — строка ввода;
- Label — надпись, метка, прикрепленная к элементу управления;
- CheckBox — флажок;
- RadioButton — радиокнопка;
- RadioGroup — группа радиокнопок, событийно связанных между собой;
- ComboBox — комбинированный список;
- ListBox — список с прокруткой.

Для создания и размещения в окне элементов диалога необходимо вызвать соответствующие функции библиотеки: txButton(), txLabel(), txEdit(), txTextEdit(), txComboBox(), txListBox(), txCheckBox(), txRadioButton(), txRadioGroup(), При этом, каждая из этих функций возвращает указатель на

объект соответствующего класса Qt. Это дает пользователю доступ ко всему набору свойств и методов таких объектов, доступных в Qt, а вместе с этим и невероятную гибкость настройки интерфейса, ограниченную лишь возможностями самого инструментария Qt. В качестве одного из обязательных параметров каждой из функций выступает пользовательская функция-обработчик — в терминах Qt — «слот», подключаемый к соответствующему сигналу объекта Qt. Это позволяет пользователю обрабатывать события, возникающие от элементов управления диалогом, например, от нажатия на кнопку, изменения текста в строке ввода, изменения состояния флажка и т. п. Более подробно это показано в таблице 3.3.

Таблица 3.3 — Элементы диалога и их описание

Объект диалога TXLib	Объект диалога QTXLib	Функция создания объекта в QTXLib	Объект класса Qt, возвращаемый функцией QTXLib	Сигнал объекта класса Qt, обрабатываемый пользователем	Описание объекта
STATIC	-		-	-	Нередактируемый элемент (картинка, текст и т. д.)
SCROLLBAR	-		-	-	Полоса прокрутки
LISTBOX	ListBox	txListBox()	QListWidget	currentRowChanged(int) — возникает при изменении текущего элемента списка	Список с прокруткой
COMBOBOX	ComboBox	txComboBox()	QComboBox	activated(int) — возникает при активации элемента	Комбинированный список
BUTTON	Button	txButton()	QPushButton	clicked() - возникает при нажатии на кнопку	Кнопка
EDIT	Edit	txEdit()	QLineEdit	EditingFinished() - возникает при окончании редактирования — нажатии Enter или потере фокуса	Строка ввода для текста

Окончание таблицы 3.3

Объект диалога TXLib	Объект диалога QTXLib	Функция создания объекта в QTXLib	Объект класса Qt, возвращаемый функцией QTXLib	Сигнал объекта класса Qt, обрабатываемый пользователем	Описание объекта
-	TextEdit	txTextEdit()	QTextEdit	textChanged() - возникает при всяком изменении текста	Многострочный редактор текста с прокруткой
-	Label	txLabel()	QLabel	-	Надпись, метка
-	CheckBox	txCheckBox()	QCheckBox	stateChanged() - возникает при изменении состояния флажка	Флажок
-	RadioButton	txRadioButton()	QRadioButton	toggled(bool) — возникает при изменении состояния кнопки	Радиокнопка
-	RadioGroup	txRadioGroup	QGroupBox	buttonClicked() - возникает при изменении состояния любой из кнопок группы	Группа радиокнопок, событийно связанных между собой

Для возможности использования механизма сигналов и слотов Qt библиотека QTXLib использует макрос Q_OBJECT и, соответственно, метаобъектный компилятор Qt MOC для своей компиляции.

Пример использования элементов диалога:

```
char buf[256];
void buttonClicked()
{
	printf("Button clicked\n");
}
void textChanged()
{
	printf("Text changed: %s\n", buf);
}
int main()
{
```

```
    txCreateWindow(270, 480, "Пример простого диалога");
    txButton(10, 10, 90, 30, "Кнопка", buttonClicked);
    txLabel(10, 50, 100, 30, "Строка ввода:");
    txEdit(10, 75, 130, 30, buf, textChanged);
    return txExec();
}
```

3.2.5 Использование определений из Qt

При разработке программ необходимо учесть, что TXLibrary использует различные WinAPI определения (констант, макросов, перечислений и т.п). Эти определения привязаны к ОС Windows и не были продублированы при реализации QTXLib, т. к. у пользователя библиотеки всегда есть возможность использовать аналогичные по своему назначению определения из пространства имен инструментария Qt. Хорошим примером может служить перечисление имен скан-кодов клавиш клавиатуры. В WinAPI такие имена имеют вид VK_XXXX, тогда как в пространстве имен Qt есть аналогичные по своему назначению определения и они имеют вид Qt::Key_XXXX (например, VK_ESCAPE и Qt::Key_Escape аналогичны по своему назначению). Дублировать такие определения нецелесообразно, поэтому пользователю предлагается использовать определения из пространства имен Qt.

3.2.6 Использование шрифтов

Еще одна важная особенность, которую необходимо обязательно учитывать при разработке программ с использованием QTXLib, не имеющая однако прямого отношения к реализации самой библиотеки как таковой — это шрифты. Названия шрифтов, используемых в функции txSelectFont могут отличаться, или же сами шрифты могут выглядеть по разному в различных операционных системах.

3.3 Архитектура библиотеки QTXLib

Графическая библиотека QTXLib реализована в виде отдельного модуля, подключаемого к программам конечного пользователя при их компиляции. Она выступает промежуточным звеном между программистом-новичком и сложными объектно-ориентированными инструментами Qt, содержит определения собственных классов для работы с графикой, окнами, диалоговыми интерфейсами, которые служат обобщением, своего рода консолидацией основных графических и интерфейсных возможностей Qt. Каждый класс созданной библиотеки для реализации своих возможностей использует отсылку к соответствующему функционалу Qt. Внешний интерфейсный уровень библиотеки, с которым непосредственно работает начинающий программист, реализован без использования объектно-ориентированного программирования и представляет собой набор обычных функций с параметрами, использование которых не представляет труда даже новичку. Вместе с тем, благодаря тесной интеграции с Qt, библиотека не ограничивает искушенного пользователя собственными функциями, предоставляя ему доступ ко всему многообразию свойств и методов объектов, доступных в Qt для еще более гибкой настройки интерфейса.

Более подробно иерархия классов графической библиотеки QTXLib показана на рисунке 3.2.

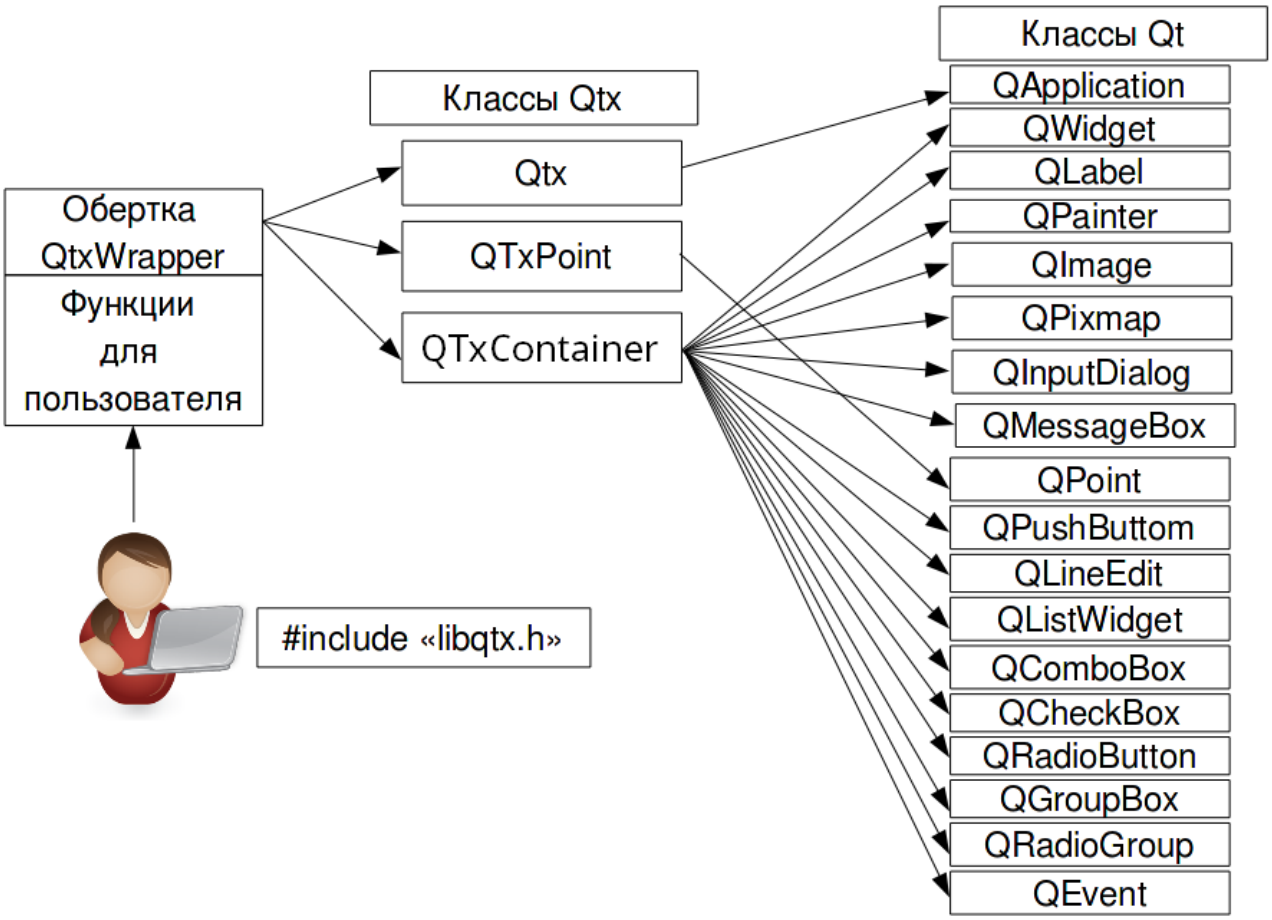

Рисунок 3.2 — Иерархия классов библиотеки QTXLib

3.4 Общая структура приложений, использующих QTXLib

Для удобства сформируем общий шаблон-структуру пользовательских приложений, использующих графическую библиотеку QTXLib в качестве средства оформления своего интерфейса.

Пользовательское приложение всегда должно включать заголовочный файл библиотеки «qtx/libqtx.h», содержать обязательную область инициализации — создание главного окна приложения с помощью функции txCreateWindow() или txInit(), а также обязательную часть запуска рабочего цикла обработки событий — вызов функции txExec(). Опционально программа может содержать область подключения собственных обработчиков событий для управления клавиатурой - с помощью функции txKeyEvent(), мышью — с помощью функции txMouseEvent(), и таймером — с помощью функции txTimerEvent(). И конечно

же должна, в таком случае, содержать и сами эти обработчики. Пример такой структуры программы показан ниже.

```
// обязательно включаем в программу заголовочный файл
#include «qtx/libqtx.h»
// опционально: функции-обработчики событий
// собственный обработчик для управления клавиатурой
void keyHandler(int key)
 {
 }
// собственный обработчик для управления мышью
void mouseHandler(int x, int y, int button, bool dbclick)
 {
 }
// собственный обработчик для событий по таймеру
void timerHandler(void *p)
 {
 }
int main()
 {
      // обязательная область инициализации — создание окна
      txCreateWindow(800, 600);
      // опциональная область подключения обработчиков
      txKeyEvent(keyHandler); // управление клавиатурой
      txMouseEvent(mouseHandler); // управление мышью
      txTimerEvent(timerHandler, NULL, 100); // и таймер
      // обязательная область запуска рабочего цикла
      return txExec();
 }
```

3.5 Компиляция библиотеки и ее использование с программами пользователей

Для работы с графической библиотекой QTXLib необходимо, чтобы в системе был установлен qt5, кроме того, нам понадобится компилятор gcc, а

также утилиты автоматической сборки проектов make и qmake.

Использование библиотеки QTXLib начинается с включения в программу пользователя директивой препроцессора #include файла qtx/libqtx.h. При этом, если компилировать и собирать программу планируется с использованием динамической библиотеки libqtx.so, т.е. без включения исходного кода библиотеки, то перед включением файла libqtx.h необходимо определить QTX_EXTERN с помощью директивы препроцессора #define.

Чтобы максимально облегчить компиляцию и сборку программ, использующих QTXLib, сделать ее максимально простой и удобной даже для новичков, совместно с библиотекой поставляется файл build — исполняемый bash-скрипт для автоматической компиляции и сборки программ. Просто запустите его на выполнение и Вы получите Вашу программу в двоичном коде готовую к запуску.

Текст файла build приведен ниже.

```
#!/bin/bash
qmake -project "QT += widgets" &&
qmake && make
rm -f moc_*.cpp && rm -f Makefile && rm -f *.pro &&
gcc *.o -shared -o libqtx.so &&
rm -f *.o
```

Первая строка сообщает системе, что файл build является исполняемым набором инструкций для командной оболочки bash.

Вторая строка создает файл проекта Qt с раширением .PRO и указывает, что проект будет использовать виджеты инструментария Qt.

Третья строка выполняет утилиту qmake которая создает специальный файл MakeFile — набор инструкций для утилиты make, которая тут же выполняет эти инструкции.

Четвертая строка удаляет файлы, созданные ранее утилитами qmake и make, которые более не нужны.

Пятая строка компонует из полученных объектных файлов динамическую библиотеку libqtx.so, которую можно испольвоть для компиляции

пользовательской программы без включения исходного кода библиотеки QTXLib.

Шестая строка удаляет ненужные более объектные файлы.

Для компиляции пользовательской программы с использованием динамической библиотеки libqtx.so (без включения исходного кода QTXLib) выполните следующий набор команд:

```
$ qmake -project "QT += widgets" "LIBS += -lqtx" && qmake && make
```

3.6 Использование среды разработки Qt Creator для создания графических приложений, использующих qtxLib

Qt Creator — современная кроссплатформенная среда разработки программного обеспечения, созданная специально для работы совместно с инструментарием Qt. Она содержит в себе удобный менеджер проектов, компилятор, отладчик, визуальный конструктор форм, профайлер и многие другие полезные возможности. Qt Creator помогает разработчику структурировать код, облегчает отладку, повышает скорость разработки больших проектов, делает написание программ более удобным и наглядным.

3.6.1 Создание библиотеки qtxLib с использованием Qt Creator

Запустим приложение Qt Creator. Нажмем кнопку «Новый проект» (рисунок 3.3) .Для создания библиотеки выберем «Библиотека C++», а затем нажмем кнопку «Выбрать».

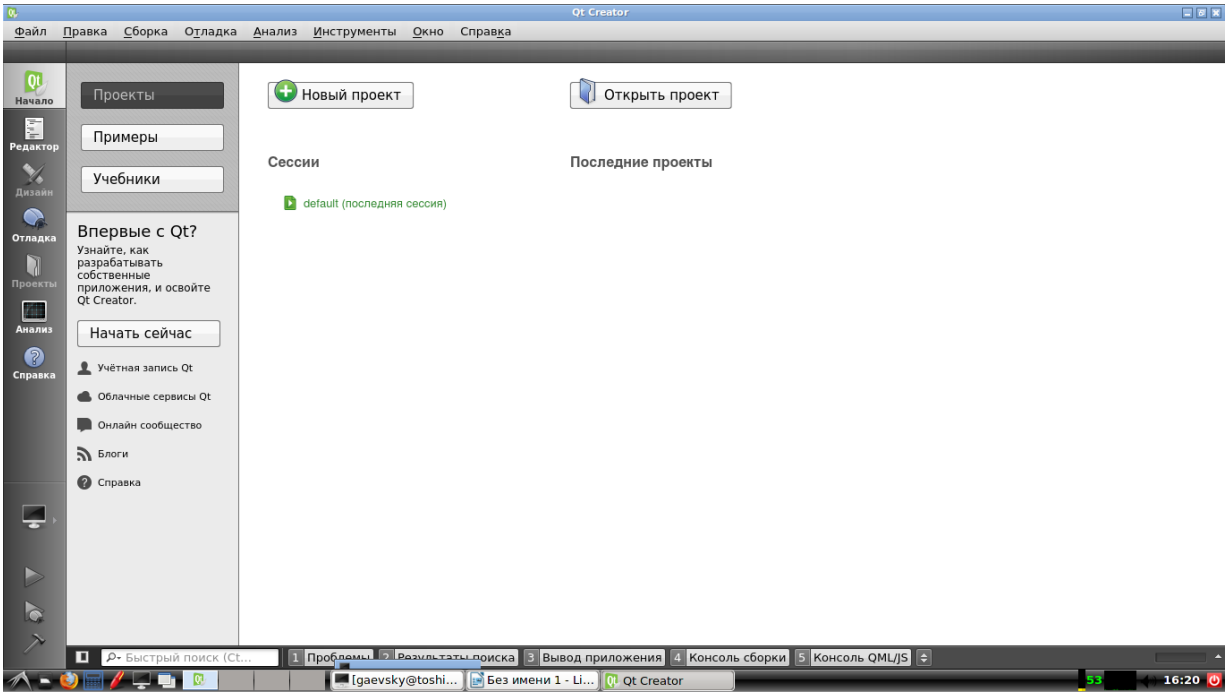

Рисунок 3.3 — Начальное окно Qt Creator после его запуска

Укажем тип создаваемой библиотеки — динамическая. Укажем название проекта и существующую папку для его размещения. Нажмем кнопку «Далее» (рисунок 3.4). Выберем комплект «Desktop» и нажмем кнопку «Далее».

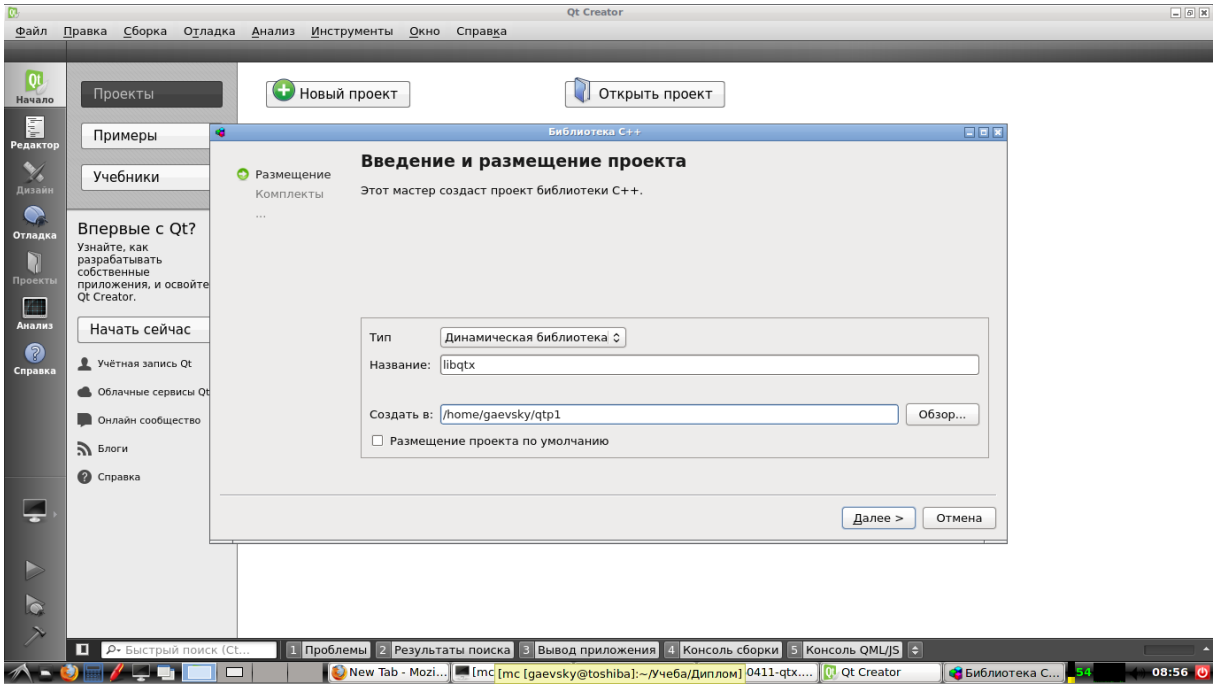

Рисунок 3.4 — Переход к созданию динамической библиотеки

Укажем необходимые модули Qt — это QtCore, QtGui и QtWidgets. Нажмем кнопку «Далее» (рисунок 3.5).

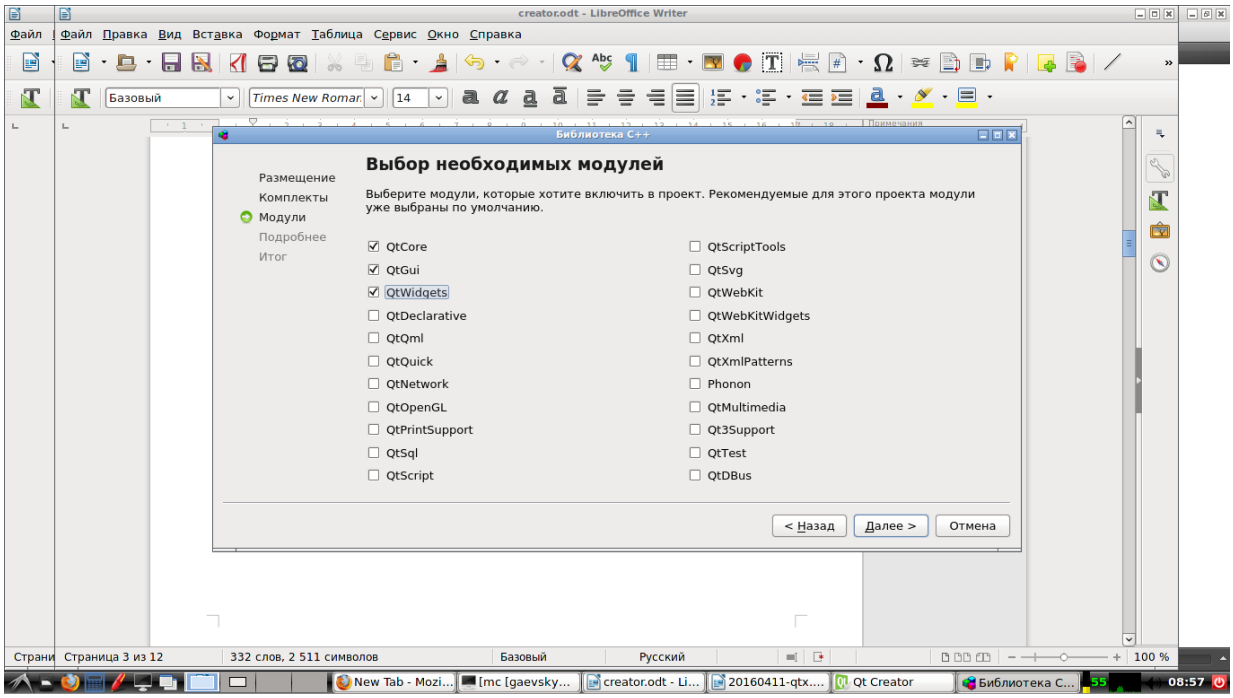

Рисунок 3.5 — Выбор модулей Qt, используемых в qtxLib

Укажем информацию об основном классе библиотеки — Qtx. Нужные имена файлов – qtx.cpp и qtx.h пропишутся автоматически (рисунок 3.6).

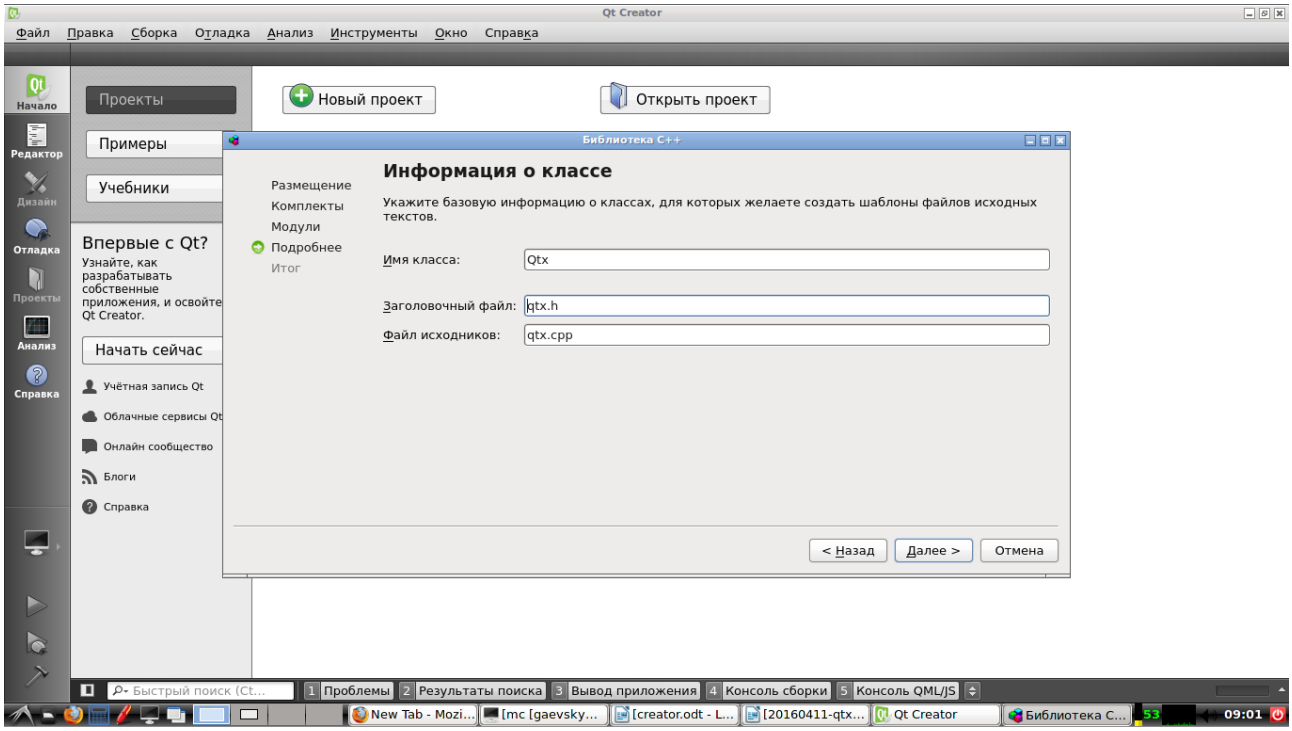

Рисунок 3.6 — Задание основного класса библиотеки

После этого нажмем кнопку «Завершить» (рисунок 3.7).

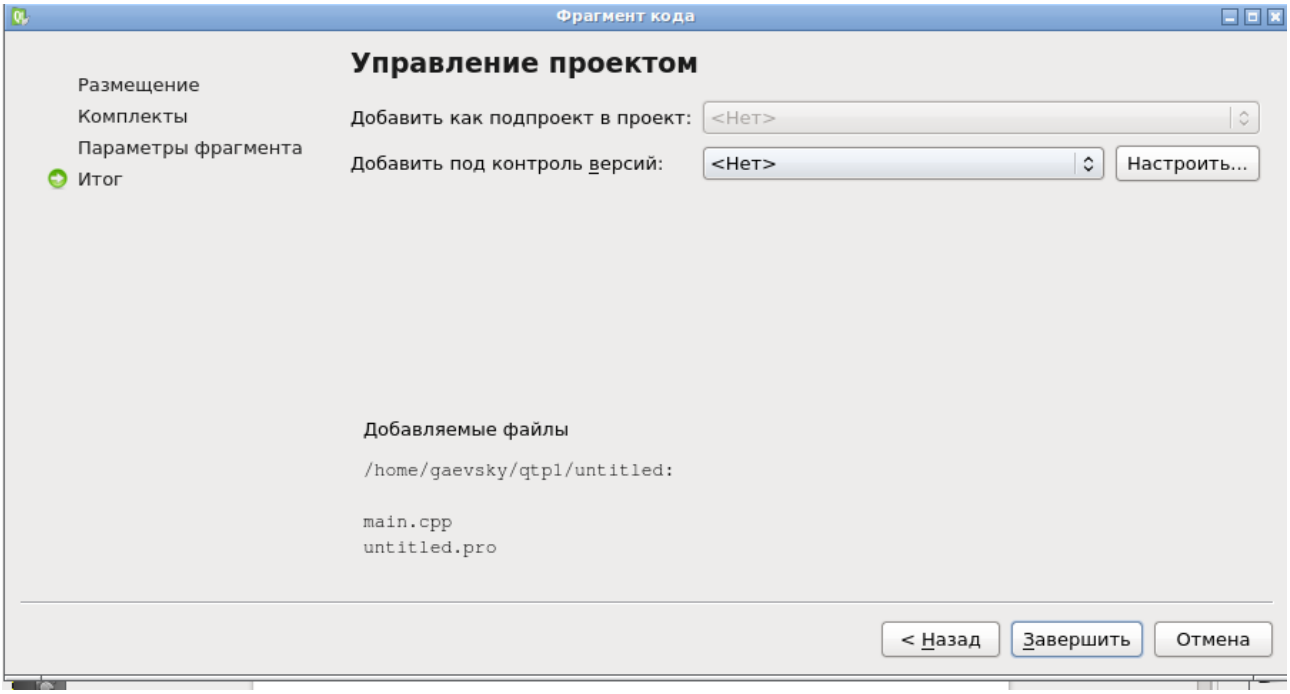

Рисунок 3.7 — Завершение формирования проекта

Перед нами откроется основная рабочая область среды Qt Creator (рисунок 3.8).

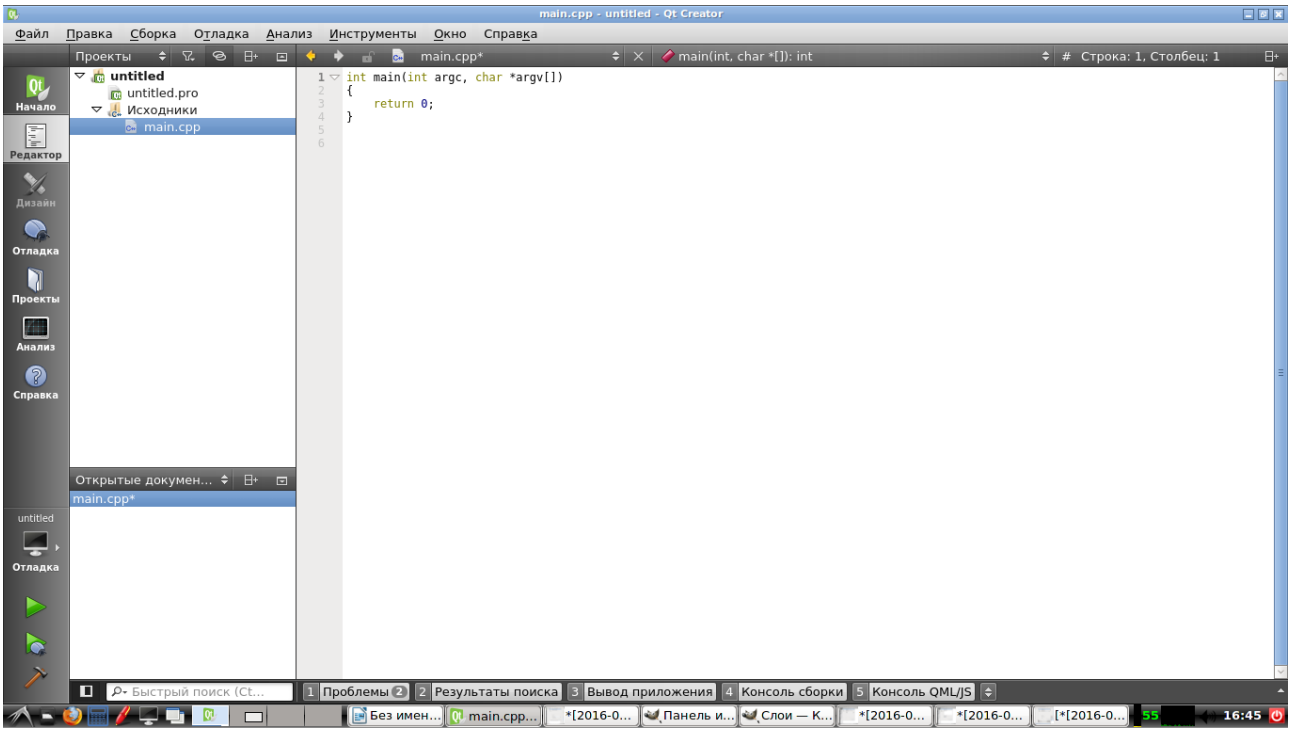

Рисунок 3.8 — Рабочая область среды Qt Creator

Щелкнем правой кнопкой мыши по названию проекта и выберем пункт меню «Добавить существующий каталог» (рисунок 3.9).

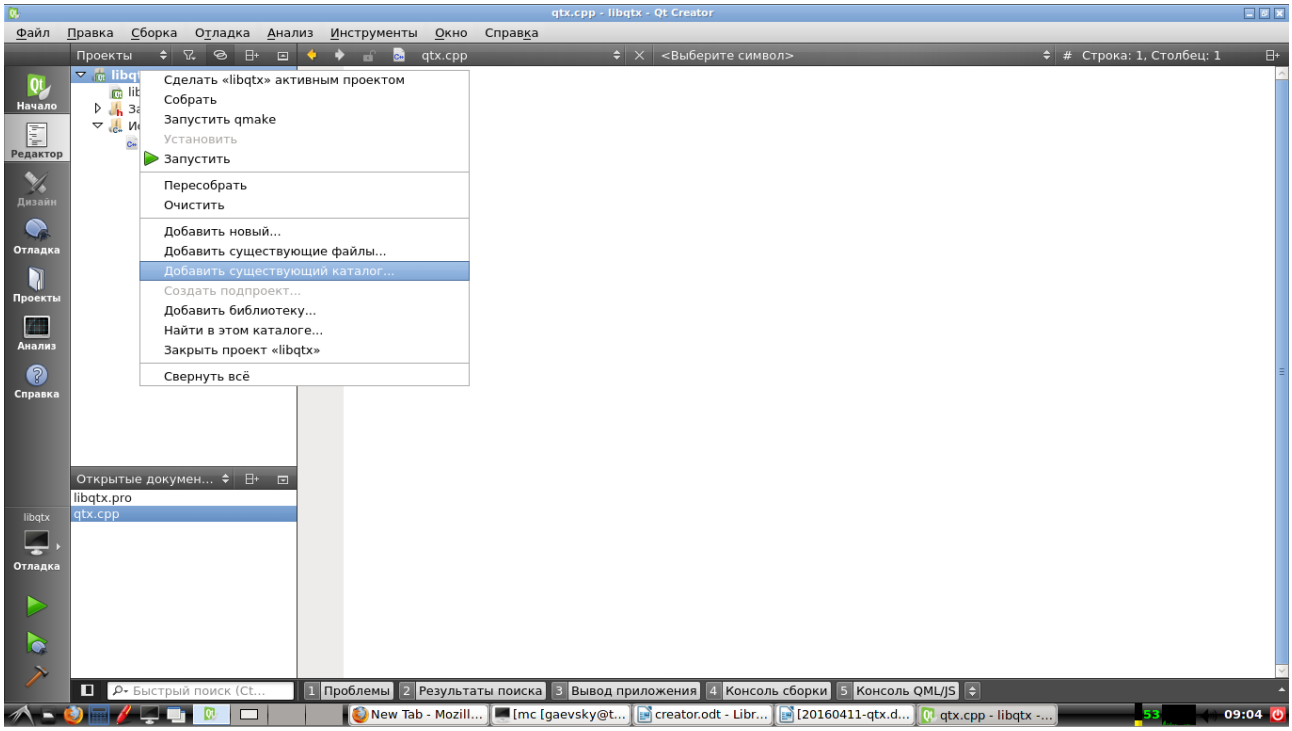

Рисунок 3.9 — Контекстное меню проекта

Нажимаем кнопку «Обзор» и указываем папку с исходными файлами библиотеки и нажимаем кнопку «Открыть» (рисунок 3.10).

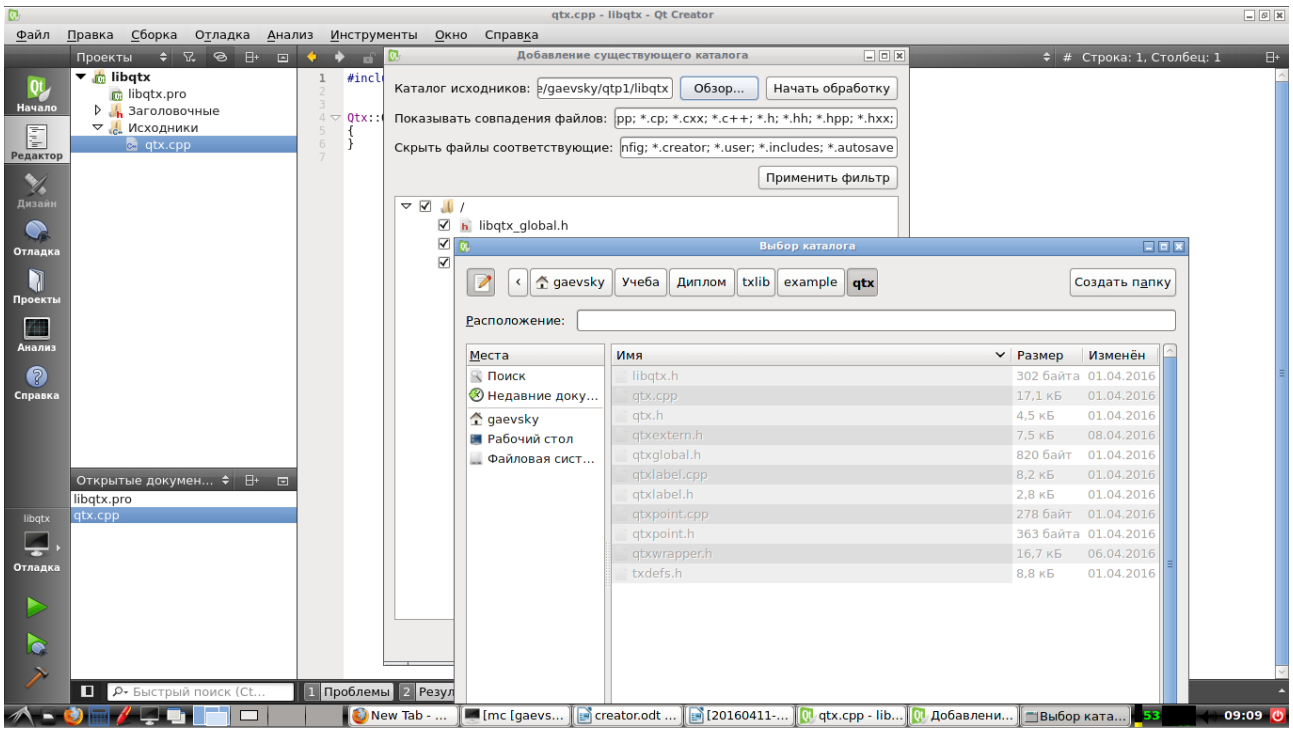

Рисунок 3.10 — Выбор папки исходных файлов библиотеки

После закрытия окна нажимаем кнопку «Начать обработку», а затем кнопку «ОК» (рисунок 3.11).

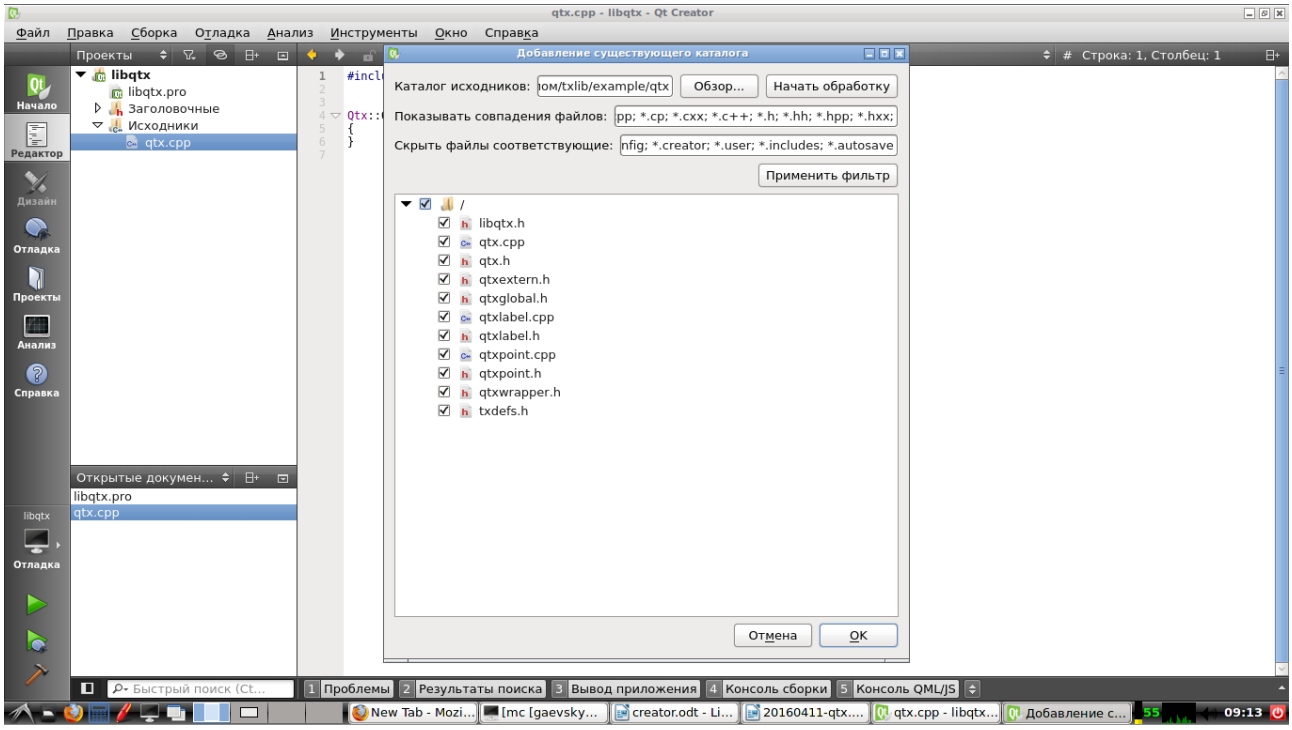

Рисунок 3.11 — Окно списка исходных файлов библиотеки

Теперь выбираем пункт меню «Сборка» - «Собрать все».

3.6.2 Создание приложения с использованием qtxLib

Запустим приложение Qt Creator (рисунок 3.12).

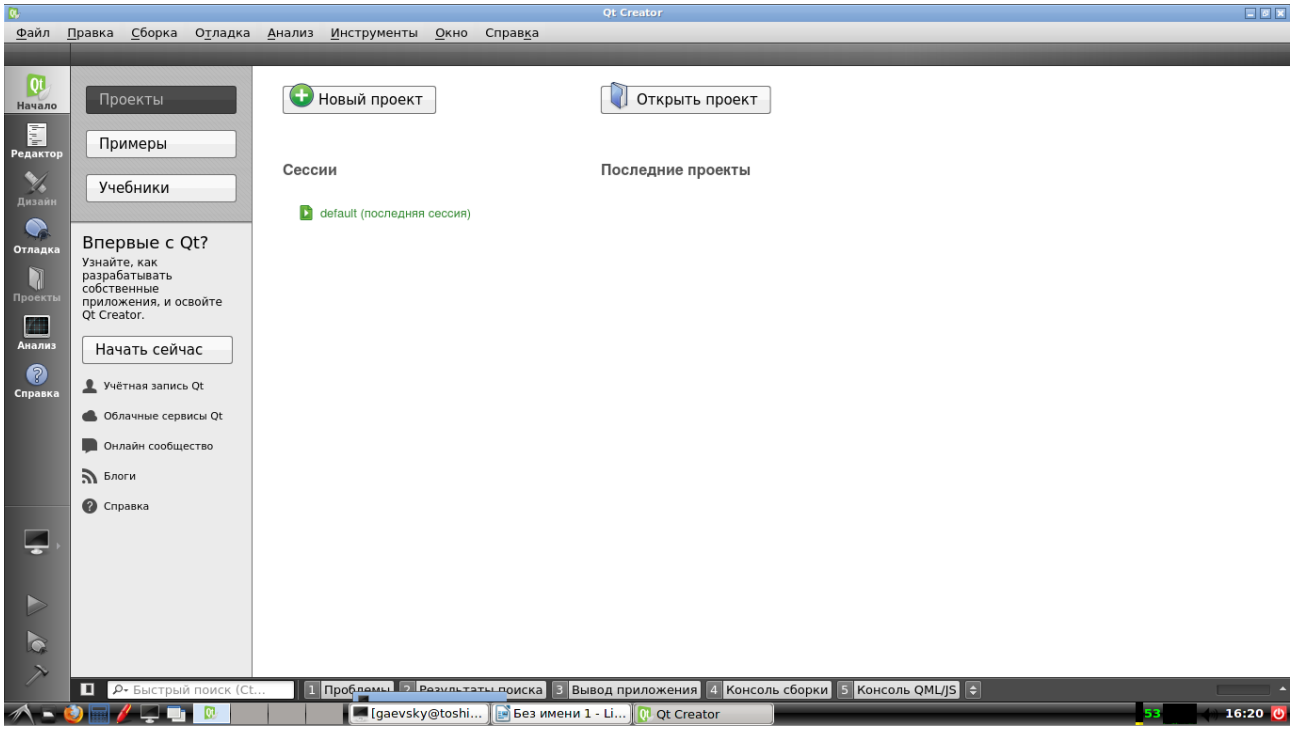

Рисунок 3.12 — Начальное окно среды Qt Creator после его запуска

Нажмем кнопку «Новый проект» (рисунок 3.13).

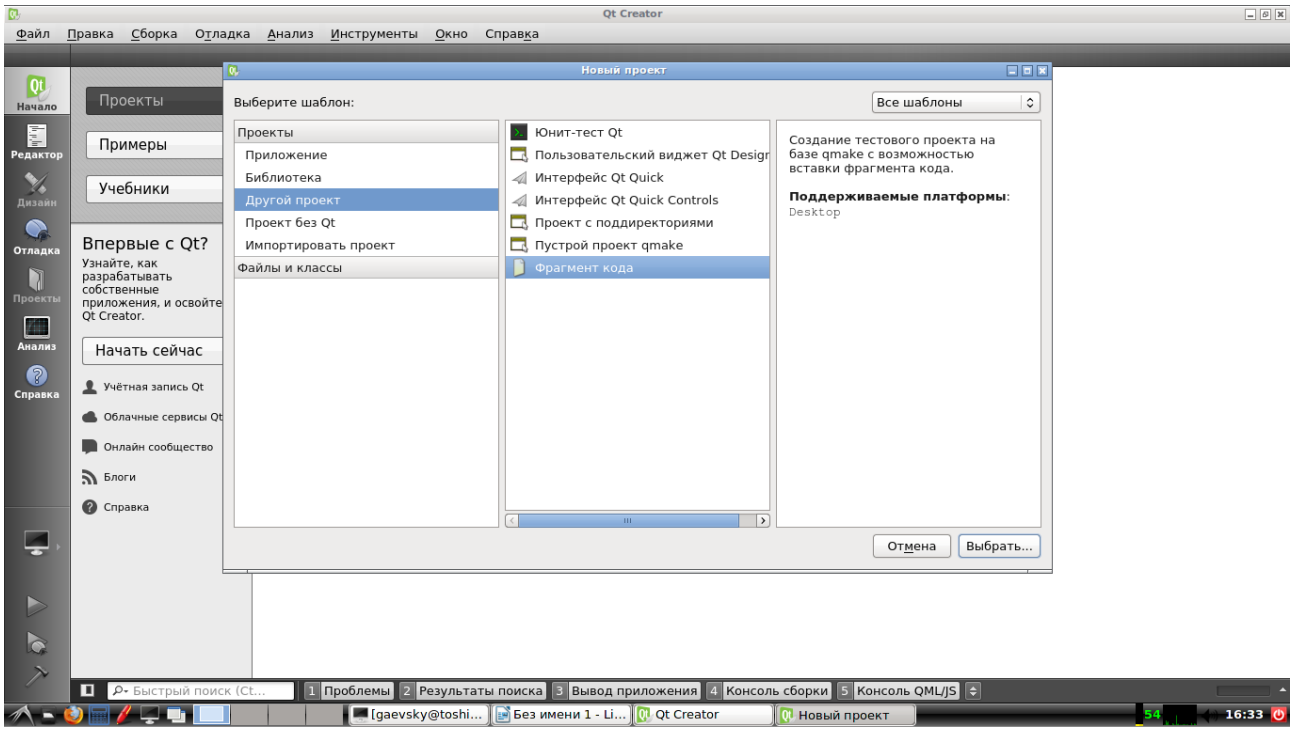

Рисунок 3.13 — Окно создания нового проекта в Qt Creator

Выберем «Другой проект» и «Фрагмент кода», а затем нажмем кнопку «Выбрать» (рисунок 3.14).

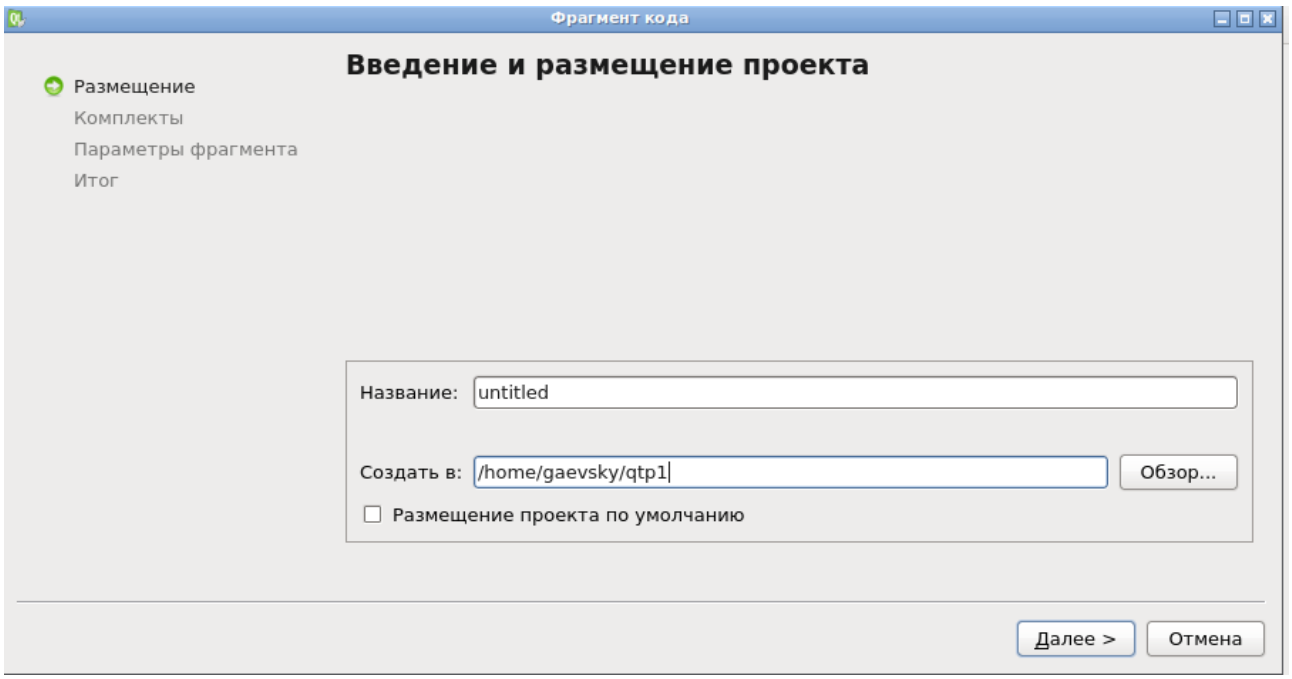

Рисунок 3.14 — Окно ввода информации о размещении проекта

Укажем название проекта и существующую папку для его размещения. Нажмем кнопку «Далее» (рисунок 3.15).

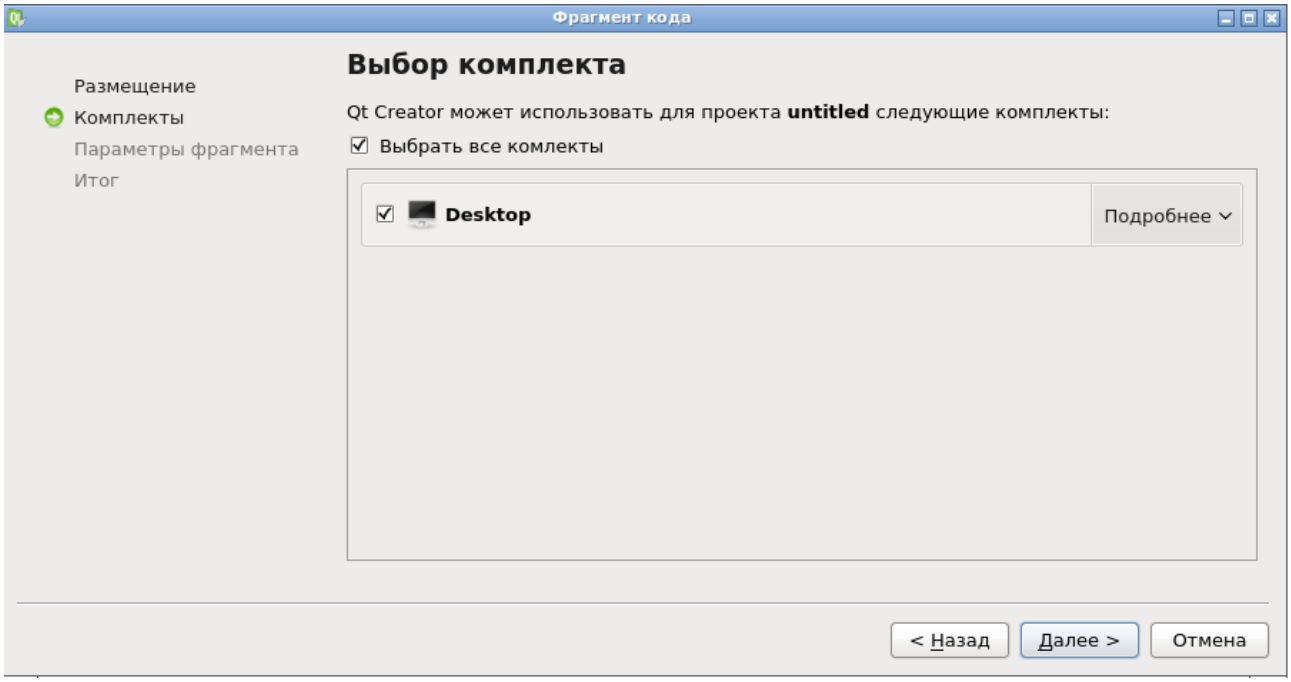

Рисунок 3.15 — Окно об указании комплекта проекта

Выберем комплект «Desktop» и нажмем кнопку «Далее» (рисунок 3.16).

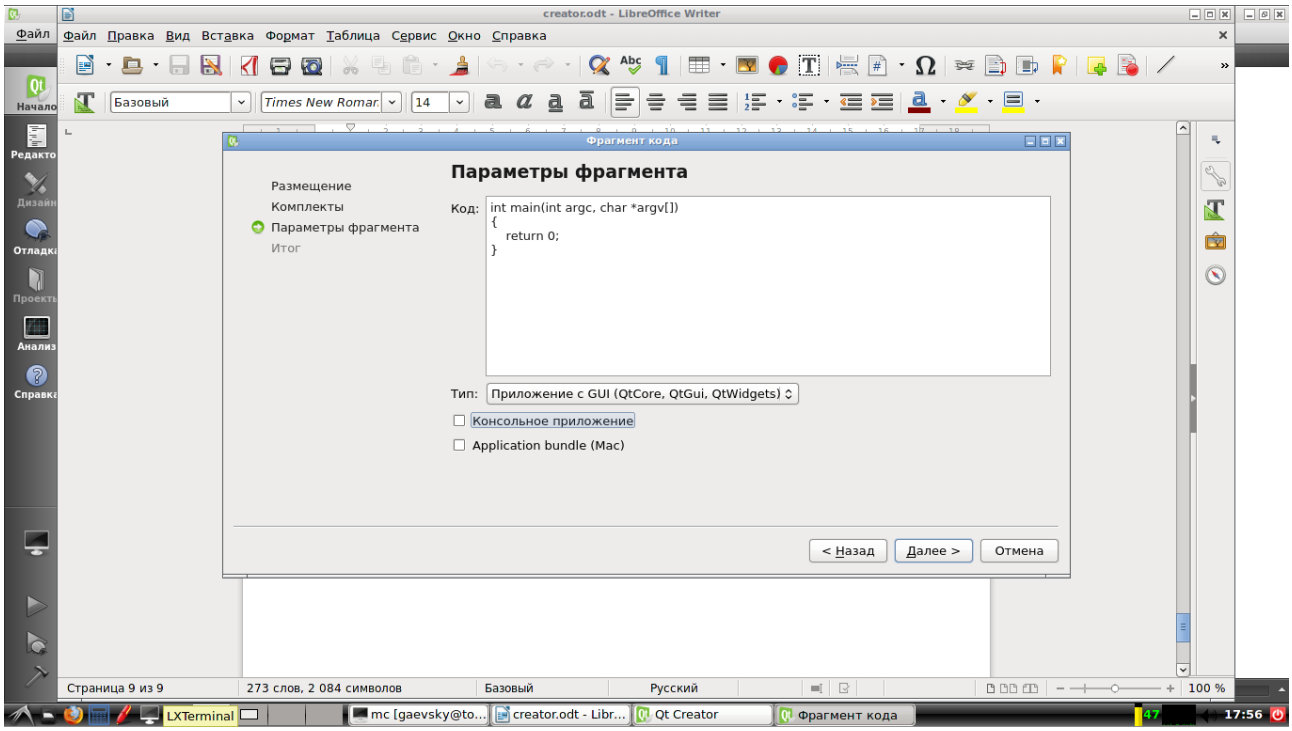

Рисунок 3.16 — Ввод информации о параметрах проекта

Укажем, что создаем приложение с GUI (QtCore, QtGui, QtWidgets) и нажмем кнопку «Далее» (рисунок 3.17).

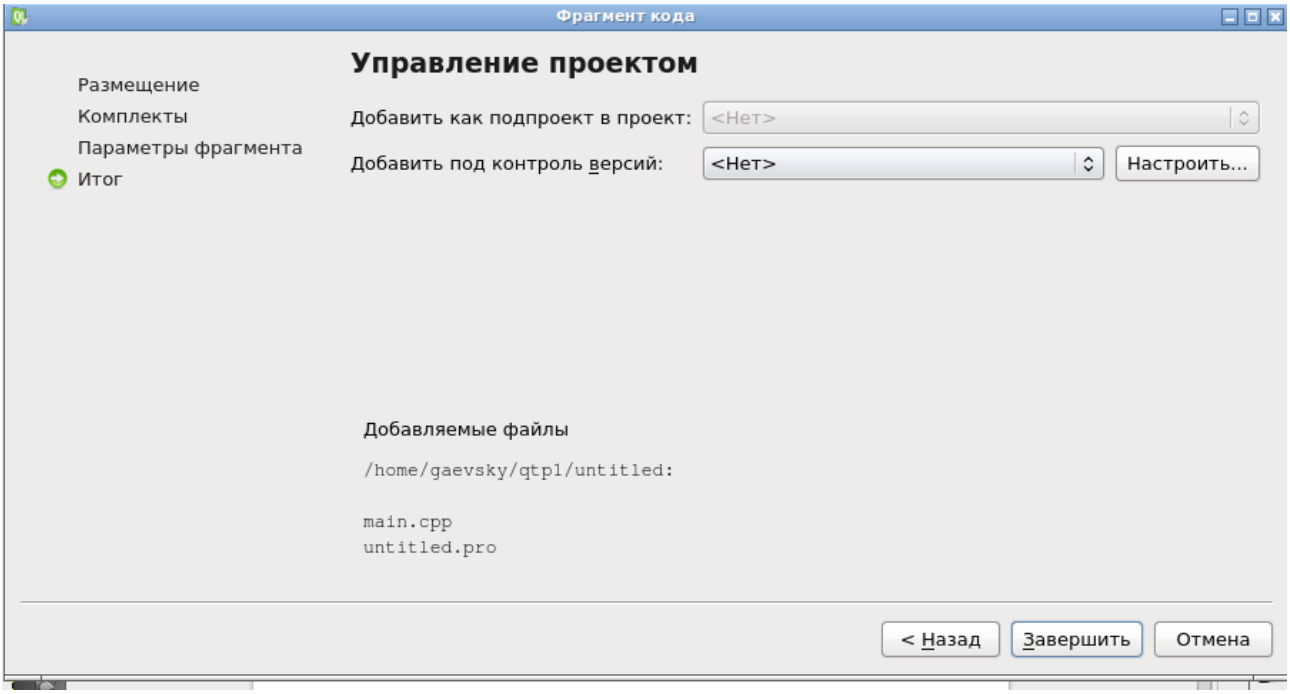

Рисунок 3.17 — Окончание создания нового проекта Qt Creator

Здесь нажмем кнопку «Завершить». Перед нами откроется основная рабочая область среды Qt Creator (рисунок 3.18).

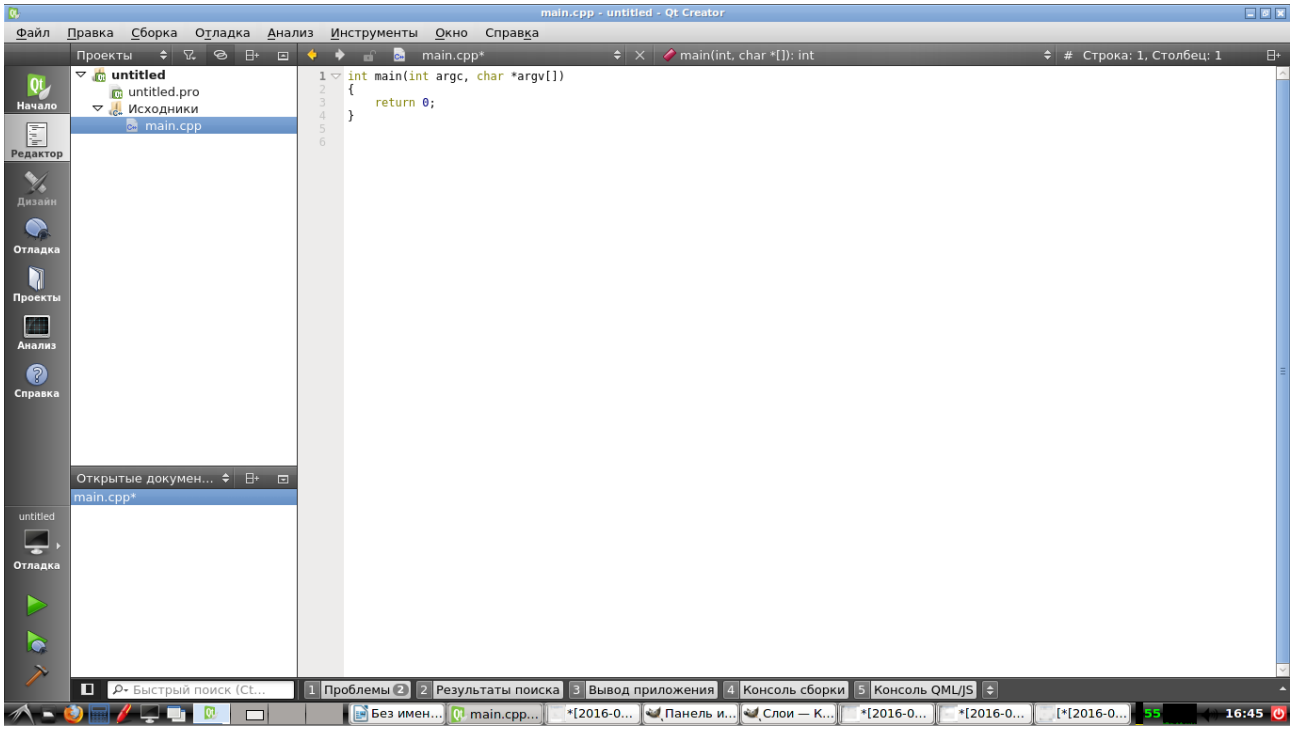

Рисунок 3.18 — Рабочая область среды Qt Creator

В диспетчере проекта щелкнем правой кнопкой мыши по имени проекта и выберем пункт «Добавить библиотеку» (рисунок 3.19).

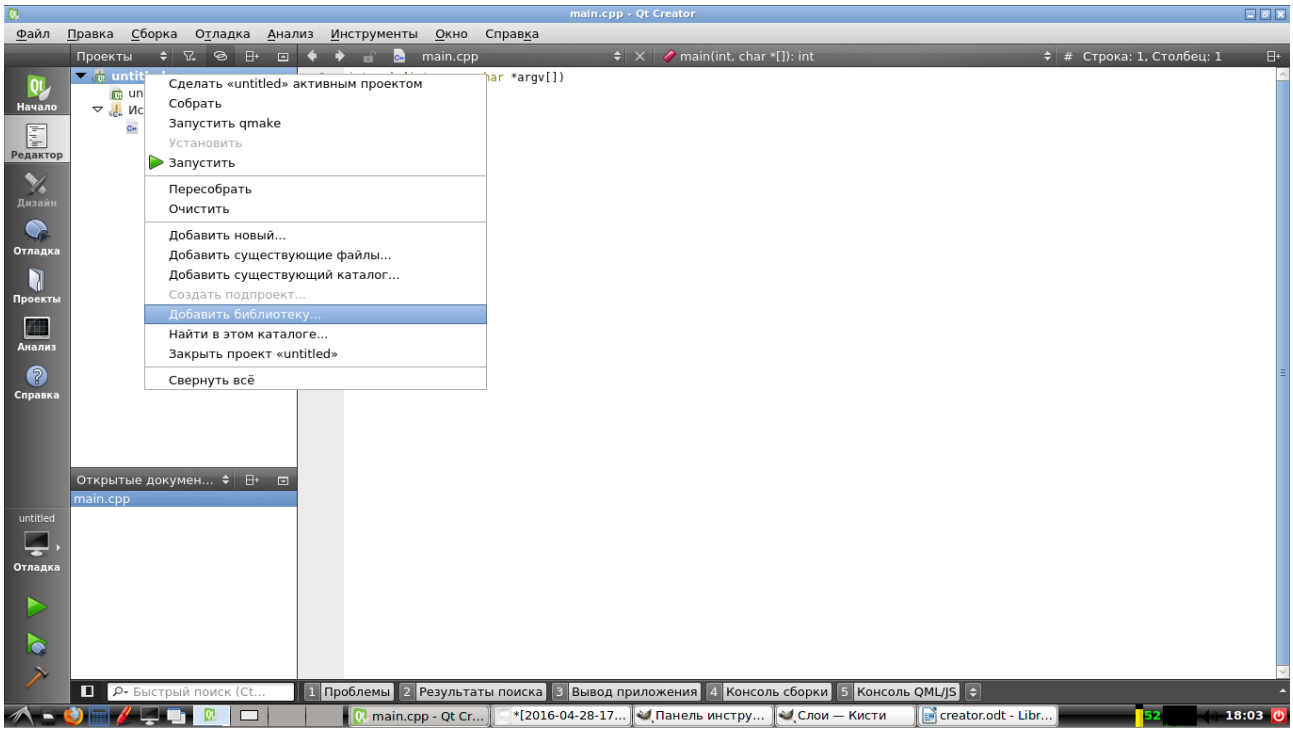

Рисунок 3.19 — Контекстное меню проекта Qt Creator

В следующем окне выберем «Внешняя» (рисунок 3.20).

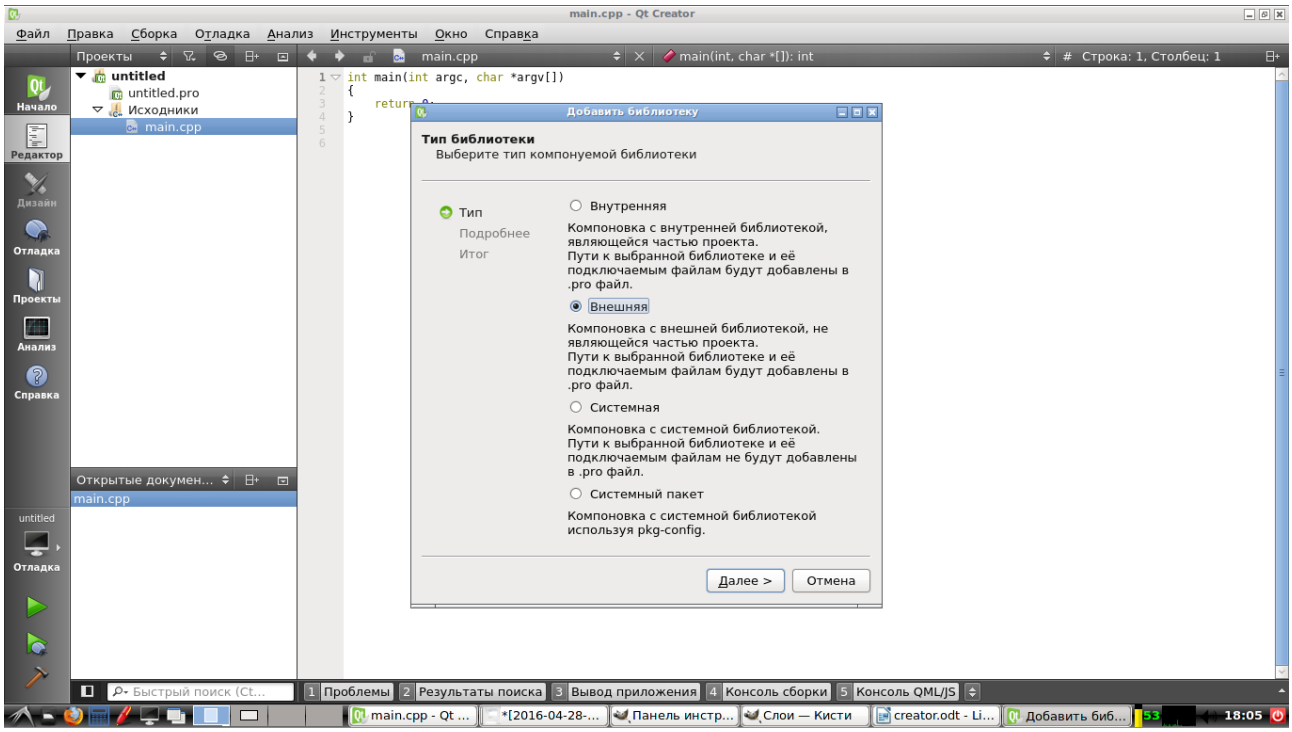

Рисунок 3.20 — Выбор типа библиотеки

В качестве файла библиотеки укажите файл libqtx.so, который находится либо в каталоге /lib64/, либо в каталоге /lib, в зависимости от разрядности операционной системы. Укажите путь к каталогу, в котором размещены заголовочные файлы библиотеки QTXLib. Обычно это каталог /usr/include/qtx (рисунок 3.21).

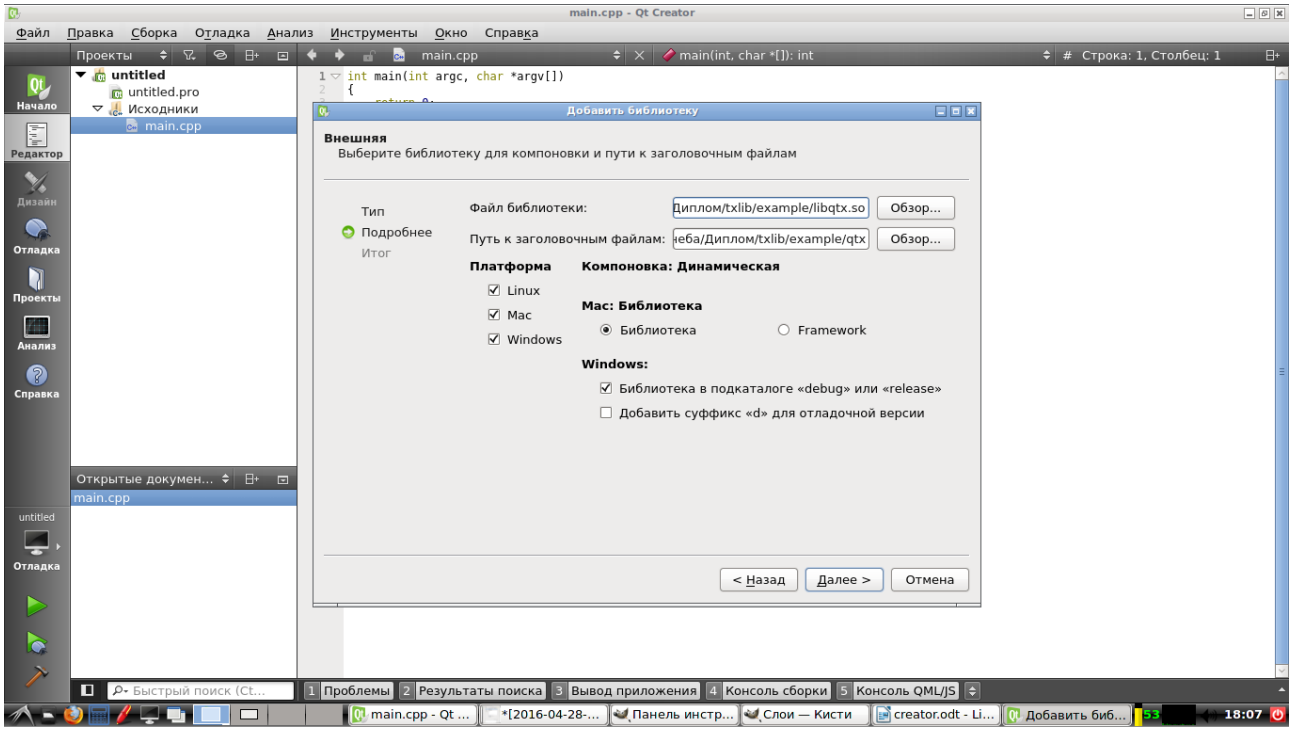

Рисунок 3.21 — Ввод сведений о размещении библиотеки

И нажмите кнопку далее. Теперь можно вернуться к файлу программы main.cpp и набрать код графической программы, не забыв про #include <qtx/libqtx.h> (рисунок 3.22).

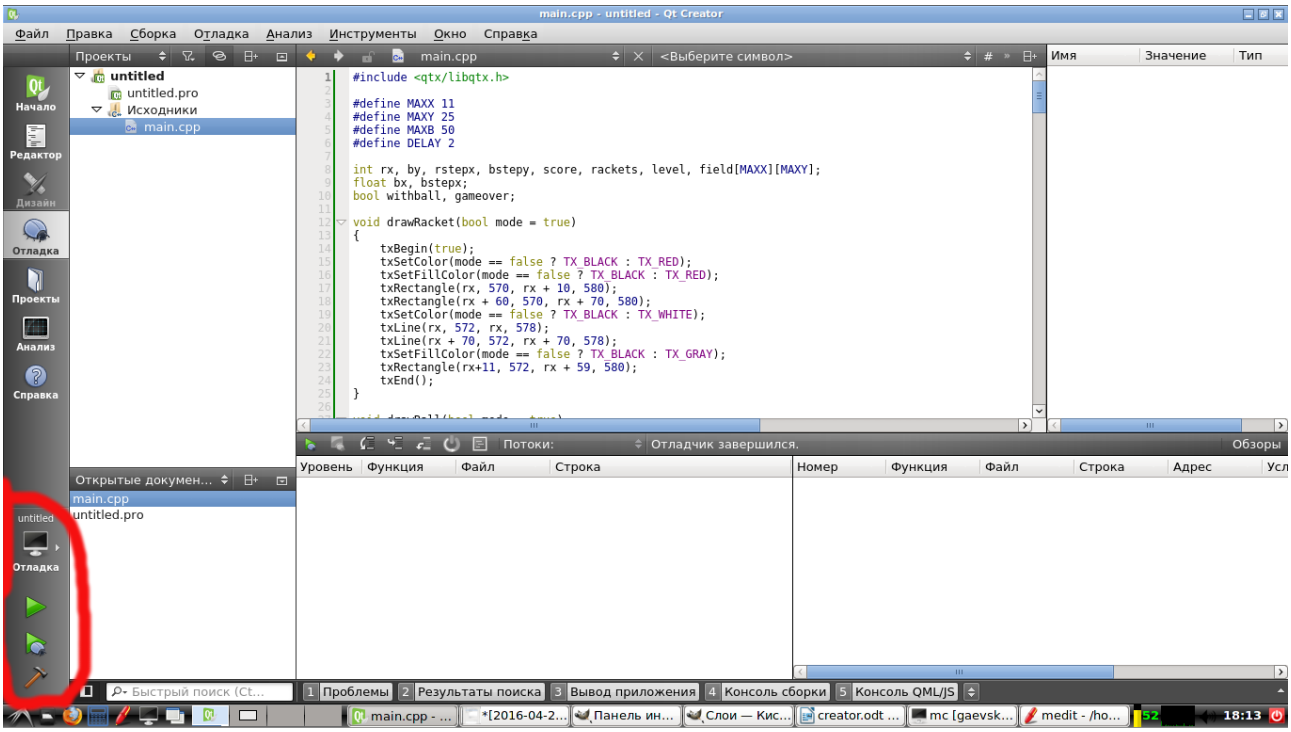

Рисунок 3.22 — программа и панель инструментов для отладки в Qt Creator

Для компиляции и запуска программы используются соответствующие элементы управления пользовательского интерфейса, расположенные на панели слева. Они выделены красным контуром (овалом).

4 Применение библиотеки QTXLib для разработки прикладных программ. Примеры использования. Перспективы развития библиотеки

С помощью графической библиотеки QTXLib можно программировать самые разные приложения, будь-то простые диалоги, программы со сложным интерфейсом или графикой, и даже игры. Работа с созданной библиотекой не предусматривает обязательного наличия глубоких знаний в программировании, в частности объектно-ориентированного подхода, ее использование просто настолько, насколько это возможно. Используя библиотеку, вы можете легко совмещать графику и элементы управления диалогом в едином пространстве, а также работать одновременно со множеством окон. Создавать диалоговые окна с ее помощью предельно просто — для этого в ней реализованы все основные элементы управления. Использовать собственные обработчики событий от таймера, мыши и клавиатуры теперь не представляет труда даже новичку.

Одновременно с простотой использования сохранена вся гибкость настройки интерфейса так необходимая профессионалам — при желании пользователю доступно множество свойств и методов, реализованных для этих целей в Qt. Программировать графику в linux теперь одно удовольствие — совместно с библиотекой поставляется скрипт для автоматической компиляции пользовательских программ, что позволяет выполнять их сборку всего в один клик.

Проект открыт для своего дальнейшего развития, в будущем в него будут добавлен новый полезный функционал:

- дополнительные элементы диалога — главное и контекстно меню, панель прогресса и многие другие;
- средства автоматического размещения и компоновки элементов диалога (layouts);
- простая возможность реализации многопоточных вычислений, в том числе ее применение в алгоритмах отрисовки самой библиотеки;
- элементы трехмерной графики с возможностью осуществления

различных преобразований;

- построение диалоговых окон на основе QML;
- возможность воспроизведения звуковых эффектов.
- простой шаблон проекта в Qt Creator для быстрого создания приложения с использованием библиотеки.

Для демонстрации основных возможностей библиотеки были специально разработаны несколько программ-примеров:

- программа, демонстрирующая работу с диалогами и простейшей графикой;
- программа — калькулятор;
- графическая игра «Питон»;
- графическая игра «Арканоид».

Скриншоты программ-примеров приведены в приложении Г.

Кроме того, адаптирована большая часть примеров, разработанная Ильей Дединским для его библиотеки TXLibrary, предназначенных для поэтапного обучения студентов программированию [1]. Все приведенные примеры находятся в архиве examples.zip и поставляются совместно с библиотекой QTXLib.

ЗАКЛЮЧЕНИЕ

В рамках выпускной квалификационной работы создана переносимая библиотека, сочетающая разнообразие возможностей по созданию простых графических приложений и простоту использования, что позволяет применять ее на начальных этапах обучения программированию.

Для достижения поставленной цели в ходе выполнения работы были решены следующие задачи:

- на основе изучения и анализа существующих графических библиотек проведен выбор средств, используемых для реализации проекта;
- изучены особенности библиотеки TXLib;
- проведено согласование интерфейса библиотеки TXLib с интерфейсом библиотеки Qt, выбранной для разработки в ходе проведенного анализа;
- сформирована концепция (стратегия) по реализации библиотеки с использованием Qt;
- реализована переносимая библиотека QTXLib, написанная на базе кроссплатформенного инструментария Qt;
- подробно описана ее архитектура и особенности реализации, рассмотрено ее подключение к программам пользователя;
- рассмотрены подробные примеры использования созданной библиотеки и возможные перспективы ее дальнейшего развития - включение в библиотеку дополнительных полезных функций - элементов диалога, средств автоматического размещения и компоновки элементов (layouts), простую возможность реализации многопоточных вычислений и многие другие.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. The Dumb Artist Library, (TX Library, TXLib) [Электронный ресурс] : описание графической библиотеки TXLib. — Режим доступа: http://ded32.net.ru/load/1-1-0-4

2. Qt документация [Электронный ресурс] : документация инструментария Qt. - Режим доступа: http://doc.crossplatform.ru/qt/

3. Официальный сайт инструментария Juce [Электронный ресурс] : документация по библиотеке Juce. - Режим доступа: http://www.juce.com/

4. Официальный сайт графической библиотеки Fox toolkit [Электронный ресурс] : документация по библиотеке Fox toolkit. - Режим доступа: http://www.fox-toolkit.org/

5. Официальный сайт графической библиотеки FLTK [Электронный ресурс] : документация по библиотеке FLTK. - Режим доступа: http://www.fltk.org/

6. Официальный сайт инструментария wxWidgets [Электронный ресурс] : документация по библиотеке wxWidgets. - Режим доступа: https://www.wxwidgets.org/

7. Официальный сайт проекта GTK+ [Электронный ресурс] : документация по библиотеке GTK+. - Режим доступа: http://www.gtk.org/

8. Русскоязычный форум инструментария Qt [Электронный ресурс] : документация по библиотеке Qt. - Режим доступа: http://www.prog.org.ru/index.php?action=forum

9. Особенности Qt [Электронный ресурс] : особенности практического применения. - Режим доступа: http://cppstudio.com/post/11167/

10. Система рисования Qt [Электронный ресурс] : полный обзор - Режим доступа: http://doc.crossplatform.ru/qt/4.5.0/paintsystem.html

11. Обзор графической библиотеки Juce [Электронный ресурс] : документация по библиотеке Juce. - Режим доступа: https://ru.wikipedia.org/wiki/Juce

12. Обзор графической библиотеки Fox Toolkit [Электронный ресурс] : документация по библиотеке Fox toolkit. - Режим доступа: https://ru.wikipedia.org/wiki/FOX_toolkit

13. Обзор графической библиотеки FLTK [Электронный ресурс] : документация по библиотеке FLTK. Режим доступа: https://ru.wikipedia.org/wiki/FLTK

14. JUCE — Кроссплатформенный C++ фреймворк для разработки приложений с пользовательским интерфейсом [Электронный ресурс] : документация по библиотеке JUCE. - Режим доступа: http://habrahabr.ru/post/209956/

15. Stroustrup, B., Programming principles and practice using C++, 2nd edition, «Pearson Education, Inc.», 2014 – 1272 с.

16. Документация по библиотеке wxWidgets [Электронный ресурс] : документация по библиотеке wxWidgets. - Режим доступа: http://www.doc.crossplatform.ru/wxwidgets/2.8.9/

17. Бланшет, С., Qt4 Программирование GUI на C++ - 2-ая ред., «КУДИЦ-пресс», 2008 – 892 с.

18. Боровский, А. Qt4.7+. Практическое программирование на C++, «БХВ-Петербург», 2012 – 496 с.

19. Саммерфилд, М., Qt Профессиональное программирование, «Символ-плюс», 2011 – 745 с.

20. Шлее, М. Qt 5.3. Профессиональное программирование на C++ (в подлиннике), «БХВ-Петербург», 2015 — 928 с.

21. Шлее, М. Профессиональное программирование на C++. +CD. Qt 4.8 (в подлиннике), «БХВ-Петербург», 2012 – 912 с.

22. Разделяемые библиотеки в linux [Электронный ресурс] : сборка и использование статических и динамических библиотек в linux — Режим доступа: https://www.opennet.ru/base/dev/shared_lib_intro.txt.html.

23. Shared libraries with GCC on Linux, [Электронный ресурс] : статья — Режим доступа: http://www.cprogramming.com/tutorial/shared-libraries-gcc.html

24. Организация исходников C++ [Электронный ресурс] : краткий обзор — Режим доступа: https://toster.ru/q/12136

25. Структура и организация проектов C++ [Электронный ресурс] : практическое применение организации сложных проектов — Режим доступа: http://eao197.narod.ru/desc/prj_struct.htm

26. Структура исходников сложного проекта C++ [Электронный ресурс]: обзор практического использования в работе — Режим доступа: https://www.linux.org.ru/forum/development/7560710

27. Официальный сайт инструментария Qt [Электронный ресурс]: домашняя страница проекта — Режим доступа: http://qt.io/ru

28. Официальный сайт Skype [Электронный ресурс]: домашняя страница проекта — Режим доступа: https://skype.com

29. Официальный сайт VirtualBox [Электронный ресурс]: домашняя страница проекта — Режим доступа: https://virtualbox.com

30. Официальный сайт KDE [Электронный ресурс]: домашняя страница проекта — Режим доступа: https://kde.org

31. Использование мета-объектного компилятора [Электронный ресурс]: QT Meta Object Compiler (MOC) — Режим доступа: http://doc.crossplatform.ru/qt/4.6.x/moc.html

32. Сайт Б. Страуструпа - американского профессора компьютерных наук, автора многих книг по программированию на C++ [Электронный ресурс]: домашняя страница — Режим доступа: http://www.stroustrup.com/

33. Официальный сайт FileZilla [Электронный ресурс]: домашняя страница проекта — Режим доступа: https://filezilla-project.org

34. Официальный сайт Audacity [Электронный ресурс]: домашняя страница проекта — Режим доступа: https://audacityteam.org

35. Официальный сайт Bittorent [Электронный ресурс]: домашняя страница проекта — Режим доступа: http://bittorent.com

36. Официальный сайт GNOME [Электронный ресурс]: домашняя страница проекта — Режим доступа: https://gnome.org

ПРИЛОЖЕНИЕ А

Функции, константы и макросы библиотеки TXLib, их краткое описание

Таблица А.1 - Функции библиотеки TXLib

Функции для рисования. Инициализация библиотеки		
№ п/п	Прототип функции / определение	Описание назначения
1	HWND txCreateWindow (double sizeX, double sizeY, bool centered=true)	Создание окна рисования.
2	bool txSetDefaults ()	Установка параметров рисования по умолчанию.
3	bool txOK ()	Проверка правильности работы библиотеки .
4	POINT txGetExtent ()	Возвращает размер окна рисования в виде структуры POINT.
5	int txGetExtentX ()	Возвращает ширину окна рисования.
6	int txGetExtentY ()	Возвращает высоту окна рисования.
7	HDC & txDC ()	Возвращает дескриптор контекста рисования холста.
8	HWND txWindow ()	Возвращает дескриптор окна холста.
9	const char * txVersion ()	Возвращает строку с информацией о текущей версии библиотеки.
10	unsigned txVersionNumber ()	Возвращает номер версии библиотеки.
11	const char * txGetModuleFileName (bool fileNameOnly=true)	Возвращает имя исполняемого файла или изначальный заголовок окна TXLib.
Функции для рисования. Установка цветов и режимов рисования		
1	COLORREF RGB (int red, int green, int blue)	Создает (смешивает) цвет из трех базовых цветов (компонент).
2	bool txSetColor (COLORREF color, double thickness=1)	Устанавливает текущий цвет и толщину линий, цвет текста.
3	bool txColor (double red, double green, double blue)	Устанавливает текущий цвет линий и текста.
4	COLORREF txGetColor ()	Возвращает текущий цвет линий и текста.
5	bool txSetFillColor (COLORREF color)	Устанавливает текущий цвет заполнения фигур.
6	bool txFillColor (double red, double green, double blue)	Устанавливает текущий цвет заполнения фигур.
7	COLORREF txGetFillColor()	Возвращает текущий цвет заполнения фигур.
8	bool txSetROP2 (int mode=R2_COPYPEN)	Устанавливает режим взаимодействия цветов при рисовании.

Продолжение таблицы А.1

№ п/п	Прототип функции / определение	Описание назначения
9	int txExtractColor (COLORREF color, COLORREF component)	Извлекает цветовую компоненту (цветовой канал) из смешанного цвета.
10	COLORREF txRGB2HSL (COLORREF rgbColor)	Преобразует цвет из формата RGB в формат HSL.
11	COLORREF txHSL2RGB (COLORREF hslColor)	Преобразует цвет из формата HSL в формат RGB.
Функции для рисования. Рисование фигур		
1	bool txClear ()	Стирает холст текущим цветом заполнения.
2	bool txSetPixel (double x, double y, COLORREF color)	Рисует пиксель (точку на экране).
3	bool txPixel (double x, double y, double red, double green, double blue)	Рисует пиксель (точку на экране).
4	COLORREF txGetPixel (double x, double y)	Возвращает текущий цвет точки (пикселя) на экране.
5	bool txLine (double x0, double y0, double x1, double y1)	Рисует линию.
6	bool txRectangle (double x0, double y0, double x1, double y1)	Рисует прямоугольник.
7	bool txPolygon (const POINT points[], int numPoints)	Рисует ломаную линию или многоугольник.
8	bool txEllipse (double x0, double y0, double x1, double y1)	Рисует эллипс.
9	bool txCircle (double x, double y, double r)	Рисует окружность или круг.
10	bool txArc (double x0, double y0, double x1, double y1, double startAngle, double totalAngle)	Рисует дугу эллипса.
11	bool txPie (double x0, double y0, double x1, double y1, double startAngle, double totalAngle)	Рисует сектор эллипса.
12	bool txChord (double x0, double y0, double x1, double y1, double startAngle, double totalAngle)	Рисует хорду эллипса.
13	bool txFloodFill (double x, double y, COLORREF color=TX_TRANSPARENT, DWORD mode=FLOODFILLSURFACE)	Заливает произвольный контур текущим цветом заполнения.

Продолжение таблицы А.1

№ п/п	Прототип функции / определение	Описание назначения
14	bool txTriangle (double x1, double y1, double x2, double y2, double x3, double y3)	Функция, которая должна бы рисовать треугольник.
15	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.
Функции для рисования. Работа с текстом		
1	bool txTextOut (double x, double y, const char text[])	Рисует текст.
2	bool txDrawText (double x0, double y0, double x1, double y1, const char text[], unsigned format=DT_CENTER|DT_VCENTER|DT_WORDBREAK|DT_WORD_ELLIPSIS)	Рисует текст, размещенный в прямоугольной области.
3	bool 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)	Выбирает текущий шрифт.
4	SIZE txGetTextExtent (const char text[])	Вычисляет размеры текстовой надписи.
5	int txGetTextExtentX (const char text[])	Вычисляет ширину текстовой надписи.
6	int txGetTextExtentY (const char text[])	Вычисляет высоту текстовой надписи.
7	unsigned txSetTextAlign (unsigned align=TA_CENTER|TA_BASELINE)	Устанавливает текущее выравнивание текста.
8	LOGFONT * txFontExist (const char name[])	Ищет шрифт по его названию.
Функции для рисования. Рисование в памяти (на "виртуальном холсте") и загрузка изображений		
1	HDC txCreateCompatibleDC (double sizeX, double sizeY, HBITMAP bitmap=NULL)	Создает дополнительный холст (контекст рисования, Device Context, DC) в памяти.

Продолжение таблицы А.1

№ п/п	Прототип функции / определение	Описание назначения
2	HDC txLoadImage (const char filename[], unsigned imageFlags=IMAGE_BITMAP, unsigned loadFlags=LR_LOADFROMFILE)	Загружает из файла изображение в формате BMP. Делает это довольно медленно.
3	bool txDeleteDC (HDC dc)	Уничтожает холст (контекст рисования, DC) в памяти.
4	bool txBitBlt (HDC dest, double xDest, double yDest, double width, double height, HDC src, double xSrc=0, double ySrc=0, DWORD rOp=SRCCOPY)	Копирует изображение с одного холста (контекста рисования, DC) на другой.
5	bool txTransparentBlt (HDC dest, double xDest, double yDest, double width, double height, HDC src, double xSrc=0, double ySrc=0, COLORREF transColor=TX_BLACK)	Копирует изображение с одного холста (контекста рисования, DC) на другой с учетом прозрачности.
6	bool txAlphaBlend (HDC dest, double xDest, double yDest, double width, double height, HDC src, double xSrc=0, double ySrc=0, double alpha=1.0)	Копирует изображение с одного холста (контекста рисования, DC) на другой с учетом прозрачности.
Функции для рисования. Вспомогательные функции		
1	double txSleep (double time)	Задерживает выполнение программы на определенное время.
2	int txBegin ()	Блокирует обновление изображения окна, во избежание мигания.
3	int txEnd ()	Разблокирует обновление окна, заблокированное функцией txBegin().
4	bool txDestroyWindow ()	Уничтожает окно TXLib.
5	double txQueryPerformance ()	Оценивает скорость работы компьютера.
6	int txUpdateWindow (int update=true)	Разрешает или запрещает автоматическое обновление изображения в окне.
7	bool txSelectObject (HGDIOBJ obj)	Устанавливает текущий объект GDI.
8	bool txIDontWantToHaveAPauseAfterMyProgramBeforeTheWindowWillClose_AndIWillNotBeAskingWhereIsMyPicture ()	Позволяет в ряде случаев избежать задержки при закрытии окна и завершением работы программы.

Продолжение таблицы А.1

Функции для рисования. Функции консоли		
№ п/п	Прототип функции / определение	Описание назначения
1	bool txSetConsoleAttr (unsigned colors=0x07)	Устанавливает цветовые атрибуты консоли.
2	unsigned txGetConsoleAttr ()	Возвращает текущие цветовые атрибуты консоли.
3	bool txClearConsole ()	Стирает текст консоли.
4	POINT txSetConsoleCursorPos (double x, double y)	Устанавливает позицию мигающего курсора консоли.
5	POINT txGetConsoleCursorPos ()	Возвращает позицию мигающего курсора консоли.
6	POINT txGetConsoleFontSize ()	Возвращает размеры шрифта консоли.
7	bool txTextCursor (bool blink=true)	Запрещает или разрешает рисование мигающего курсора в окне.
Функции для работы с мышью		
1	POINT txMousePos ()	Возвращает позицию мыши.
2	int txMouseX ()	Возвращает X-координату мыши.
3	int txMouseY ()	Возвращает Y-координату мыши.
4	int txMouseButtons ()	Возвращает состояние кнопок мыши.
Функции, реализующие диалоговые окна		
1	const char * txInputBox (const char *text=NULL, const char *caption=NULL, const char *input=NULL)	Ввод строки в отдельном окне.
Разные прочие функции		
1	bool In (T x, T a, T b)	Проверка, находится ли параметр x внутри замкнутого интервала [a; b].
2	int random (int range)	Генератор случайных чисел
3	double random (double left, double right)	Генератор случайных чисел
4	double txSqr (double x)	Очень удобное возведение числа в квадрат.
5	void txDump (const void *address, const char name[]="txDump()")	Распечатывает дамп области памяти в консоли.
6	bool txPlaySound (const char filename[]=NULL, DWORD mode=SND_ASYNC)	Воспроизводит звуковой файл.
7	unsigned txMessageBox (const char *text, const char *header="TXLib сообщает", unsigned flags=0)	Выводит сообщение в окне с помощью функции MessageBox.

Продолжение таблицы А.1

№ п/п	Прототип функции / определение	Описание назначения
8	bool txNotifyIcon (unsigned flags, const char title[], const char format[],…)	Выводит всплывающее сообщение в системном трее.
9	int txOutputDebugPrintf (const char format[],…)	Выводит сообщение в отладчике.
10	bool In (const POINT &pt, const RECT &rect)	Проверка, находится ли точка pt внутри прямоугольника rect.
Служебные функции		
1	int txUpdateWindow (int update=true)	Разрешает или запрещает автоматическое обновление изображения в окне.
2	bool txSelectObject (HGDIOBJ obj)	Устанавливает текущий объект GDI.
3	WNDPROC txSetWindowsHook (WNDPROC wndProc=NULL)	Устанавливает альтернативную функцию обработки оконных сообщений Windows (оконную функцию) для окна TXLib.
4	bool txLock (bool wait=true)	Блокировка холста (контекста рисования).
5	bool txUnlock ()	Разблокировка холста
Класс txAutoLock		
1	txAutoLock (CRITICAL_SECTION *cs, bool mandatory=true)	Конструктор, блокирует критическую секцию
2	txAutoLock (bool mandatory=true)	Конструктор для блокировки холста TXLib.
3	~txAutoLock ()	Деструктор, разблокирует секцию
4	operator bool () const	Позволяет проверить, заблокировалась секция или нет
5	CRITICAL_SECTION * cs_	Блокируемая критическая секция
6	txAutoLock (const txAutoLock &)	Закрытый член класса
Класс txDialog		
1	TxDialog ()	Конструктор.
2	txDialog (const Layout *layout)	Конструктор.
3	virtual ~txDialog ()	Деструктор.
4	const Layout * setLayout (const Layout *layout)	Устанавливает текущий макет диалогового окна.
5	virtual int dialogProc (HWND _wnd, UINT _msg, WPARAM _wParam, LPARAM _lParam)	Функция обработки сообщений диалогового окна.
6	INT_PTR dialogBox (const Layout *layout=NULL, size_t bufsize=0)	Запускает диалоговое окно.

Окончание таблицы А.1

№ п/п	Прототип функции / определение	Описание назначения
7	INT_PTR dialogBox (WORD resource)	Запускает диалоговое окно.
8	txDialog (const txDialog &)	Закрытые конструктор копирования и оператор присваивания.
9	static ptrdiff_t CALLBACK dialogProc__ (HWND wnd, UINT msg, WPARAM wParam, LPARAM lParam)	Настоящая диалоговая функция (не txDialog::dialogProc(), т.к. функция окна in32 должна быть статической).
10	const Layout * layout_	Текущий макет диалога.
Структура txDialog::Layout		
1	CONTROL wndclass	Тип контрола
2	const char * caption	Название или текст
3	WORD id	Идентификатор контрола
4	short x	Координата верхнего левого угла
5	short y	Координата нижнего правого угла
6	short sx	Размер по X.
7	short sy	РазмеNDEBUGр по Y.
8	DWORD style	Стиль контрола
9	const char * font	Шрифт диалогового окна
10	WORD fontsize	Размер шрифта диалогового окна

Таблица А.2 - Константы и перечисления библиотеки TXLib

№ п/п	Определение	Описание назначения
1	const int _TX_TIMEOUT = 1000	Таймаут операций ожидания (мс)
2	const unsigned _TX_BUFSIZE = 1024	Размеры внутренних статических строковых буферов TXLib.
3	const unsigned _TX_BIGBUFSIZE = 2048	Размеры внутренних статических строковых буферов TXLib.
4	const double txPI	Число Пи
5	unsigned _txConsoleMode = SW_HIDE	Режим отображения консольного окна. Допустимы любые флаги функции ShowWindow.
6	unsigned _txWindowStyle = WS_POPUP | WS_BORDER | WS_CAPTION | WS_SYSMENU	Стиль графического окна библиотеки.

Продолжение таблицы А.2

№ п/п	Определение	Описание назначения
7	const char * _txConsoleFont = "Lucida Console"	Шрифт консоли
8	unsigned _txCursorBlinkInterval = 500	Интервал мигания курсора консоли (мс)
9	int _txWindowUpdateInterval = 25	Интервал обновления холста (мс)
10	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_LIGHTNESS = 0x06000000 }	Перечисление предопределенных цветов.

Окончание таблицы А.2

№ п/п	Определение	Описание назначения
11	enum CONTROL { DIALOG = 0x00000000, BUTTON = 0xFFFF0080, EDIT = 0xFFFF0081, STATIC = 0xFFFF0082, LISTBOX = 0xFFFF0083, SCROLLBAR = 0xFFFF0084, COMBOBOX = 0xFFFF0085, END = 0x00000000 }	Константы для задания типа элемента управления.

Таблица А.3 - Макросы библиотеки TXLib

№ п/п	Определение макроса	Описание назначения
1	#define MAX(a, b)	Возвращает максимальное из двух чисел
2	#define MIN(a, b)	Возвращает минимальное из двух чисел
3	#define ROUND(x)	Округляет число до целого
4	#define ZERO(type)	Обнулитель типов, не имеющих конструкторов
5	#define assert(cond)	Замена стандартного макроса assert(), с выдачей сообщения через txMessageBox(), консоль и OutputDebugString().
6	#define asserted	Выводит диагностическое сообщение в случае нулевого или ложного результата.
7	#define verify	Выполняет команду (вычисляет выражение) и проверяет результат.
8	#define TX_ERROR(…)	Выводит развернутое диагностическое сообщение.
9	#define TX_DEBUG_ERROR	Выводит развернутое диагностическое сообщение в отладочном режиме.
10	#define __TX_COMPILER__	Имя и версия текущего компилятора
11	#define __TX_FILELINE__	Макрос, раскрывающийся в имя файла и номер строки файла, где он встретился.
12	#define __TX_FUNCTION__	Имя текущей функции
13	#define _TX_BUILDMODE	Имя режима сборки
14	#define sizearr(arr)	Вычисление размера массива в элементах
15	#define TX_AUTO_FUNC(param_t, param, func)	Автоматический вызов функции при завершении другой функции (аналог __finally)
16	#define _	Макрос, позволяющий передать переменное число параметров в какой-либо другой макрос.
17	#define TX_COMMA	Синоним макроса _ (символ подчеркивания)

Окончание таблицы А.3

№ п/п	Определение	Описание назначения
18	#define _TX_MODULE	Имя модуля TXLib, входит в диагностические сообщения.
19	#define SIZEARR(arr) (sizeof (arr) / sizeof (0[arr]))	Вычисление размера массива в элементах
20	#define TX_BEGIN_MESSAGE_ MAP()	Заголовок карты сообщений (Message Map).
21	#define TX_HANDLE(id)	Заголовок обработчика сообщения (Message handler) карты сообщений.
22	#define TX_COMMAND_MAP	Начало карты команд (Command map) в карте сообщений.
23	#define TX_END_MESSAGE_M AP	Завершитель карты сообщений.
24	#define __TX_DEBUG_MACRO S	Отладочная печать переменной во время вычисления выражения или участка кода во время его выполнения
25	#define txGDI(command)	Вызов функции Win32 GDI с автоматической блокировкой и разблокировкой.
26	#define _TX_VERSION	Текущая версия библиотеки.
27	#define _TX_VER	Версия библиотеки в целочисленном формате.
28	#define _TX_NOINIT	Запрет ранней инициализации TXLib.
29	#define _TX_WAITABLE_PARE NTS	Список запускающих программ, которые ждут нажатия клавиши после завершения процесса TXLib.
30	#define _TX_ALLOW_KILL_PA RENT true	Разрешать принудительное завершение вызывающих программ, ждущих нажатия клавиш после завершения TXLib.
31	#define _TX_ALLOW_TRACE	Включает/отключает внутреннюю трассировку исполнения кода библиотеки.
32	#define TX_TRACE	Трассирует исполнение кода через OutputDebugString().

ПРИЛОЖЕНИЕ Б

Классы Qt, используемые при реализации QTXLib

Таблица Б.1 - Классы Qt для рисования

Наименование класса	Назначение класса	Где используется в QTXLib
QBitmap	Монохромное (глубина в 1 бит) растровое изображение	класс Qtx
QBrush	Определяет образец заливки фигур, отображаемых с помощью QPainter	класс Qtx
QColor	Цвета, основанные на значениях цветовых моделей RGB, HSV или CMYK	класс Qtx
QColormap	Отображает независимые от устройства цвета QColors в зависимые от устройства значения пикселей	класс Qtx
QConicalGradient	Используется в комбинации с QBrush для задания кисти с коническим градиентом	
QDirectPainter	Прямой доступ к аппаратной видеосистеме в Qt для встраиваемых Linux-систем	
QFont	Устанавливает шрифт, используемый для отрисовки текста	класс Qtx
QFontMetrics	Информация о метриках шрифта	класс Qtx
QFontMetricsF	Информация о метриках шрифта	класс Qtx
QGenericMatrix	Класс-шаблон, который представляет матрицу преобразований NxM с N столбцами и M строками	
QGradient	Используется совместно с QBrush для задания градиентной заливки	
QIcon	Масштабируемые пиктограммы для различных режимов и состояний	класс Qtx
QIconEngine	Абстрактный базовый класс для отображения QIcon	
QIconEngineV2	Абстрактный базовый класс для отображения QIcon	
QImage	Аппаратно-независимое представление изображения, предоставляющее прямой доступ к пикселям и способная работать в качестве устройства рисования	класс Qtx
QImageReader	Форматонезависимый интерфейс для чтения изображений из файлов и других устройств	
QImageWriter	Форматонезависимый интерфейс для записи изображений в файлы или другие устройства	
QLine	Двумерный вектор, использующий целочисленные значения для задания координат	

Продолжение таблицы Б.1

Наименование класса	Назначение класса	Где используется в QTXLib
QLineF	Двумерный вектор, использующий значения с плавающей точкой для задания координат	
QLinearGradient	Используется совместно с QBrush для задания заливки в виде линейного градиента	
QMargins	Определяет четыре поля прямоугольника	
QMovie	Вспомогательный класс для проигрывания роликов в QImageReader	
QPaintDevice	Базовый класс для объектов, которые могут быть отображены	класс Qtx
QPaintEngine	Абстрактное описание процесса рисования QPainter на указанном устройстве на указанной платформе	класс Qtx
QPainter	Выполняет низкоуровневое рисование на виджетах и других устройствах рисования	класс Qtx
QPainterPath	Контейнер для операций рисования, позволяющий создавать и повторно использовать графические фигуры	
QPainterPathStroker	Используется для генерирования заполненной границы для указанной траектории рисовальщика	
QPalette	Содержит цветовые группы для каждого состояния виджета	класс Qtx
QPen	Задает для QPainter способ рисования линий и контуров фигур	класс Qtx
QPicture	Устройство рисования, запоминающее и повторяющее команды QPainter	
QPixmap	Неэкранное представление изображения, которое может использоваться в качестве устройства рисования	класс Qtx
QPixmapCache	Кэш растровых изображений всего приложения	
QPoint	Описывает точку на плоскости, используя целые числа	класс QtxPoint
QPointF	Описывает точку на плоскости, используя числа с плавающей точкой	
QPolygon	Вектор точек с координатами, заданными целыми числами	класс Qtx
QPolygonF	Вектор точек с координатами, заданными числами с плавающей точкой	
QRadialGradient	Используется совместно с QBrush для задания заливки с радиальным градиентом	
QRect	Определяет прямоугольник на плоскости, использующий целые значения для задания своих координат	класс Qtx

Окончание таблицы Б.1

Наименование класса	Назначение класса	Где используется в QTXLib
QRectF	Определяет прямоугольник на плоскости, использующий для задания своих координат значения с плавающей точкой	
QRegion	Устанавливает область отсечения для рисовальщика (painter)	
QSize	Определяет размер двумерного объекта, используя целые числа	класс Qtx
QSizeF	Определяет размер двумерного объекта, используя числа с плавающей точкой	
QStylePainter	Вспомогательный класс для рисования элементов QStyle внутри виджета	
QSvgGenerator	Устройство рисования, которое используется для создания рисунков SVG	
QSvgRenderer	Используется для отображения содержимого файла SVG на устройстве рисования	
QSvgWidget	Виджет, используемый для отображения файлов масштабируемой векторной графики (Scalable Vector Graphics, SVG)	
QTransform	Задает двухмерные преобразования системы координат	класс Qtx
QVector2D	Представляет вектор или вершину (vertex) в 2D пространстве	

Таблица Б.2 - Классы основных виджетов Qt

Наименование класса	Назначение класса	Где используется в QTXLib
QCheckBox	Флажок с текстовой меткой	класс QTxContainer
QComboBox	Сочетание кнопки и всплывающего списка	класс QTxContainer
QCommandLinkButton	Командная кнопка-ссылка в стиле Vista	класс QTxContainer
QDateEdit	Виджет для редактирования даты, основанный на виджете QDateTimeEdit	
QDateTimeEdit	Виджет для редактирования даты и времени	
QDial	Элемент управления, проградуированный по окружности (наподобие спидометра или потенциомера)	
QDoubleSpinBox	Виджет счетчика, принимающего значения с плавающей точкой	
QFocusFrame	Рамка фокуса, которая может отображаться вокруг стандартной области прорисовки виджета	
QFontComboBox	Выпадающий список, который позволяет пользователю выбрать семейство шрифтов	

Окончание таблицы Б.2

Наименование класса	Назначение класса	Где используется в QTXLib
QLCDNumber	Отображает число с помощью цифр, имитирующих ЖК-индикатор	
QLabel	Отображает текст или рисунок	класс QTxContainer
QLineEdit	Однострочный редактор текста	класс QTxContainer
QMenu	Виджет меню, используемый в панели меню, контекстном меню и других всплывающих меню	
QProgressBar	Горизонтальный или вертикальный индикатор выполнения	
QPushButton	Командная кнопка	класс QTxContainer
QRadioButton	Радио-кнопка с текстовой меткой	класс QTxContainer
QScrollArea	Область прокрутки на другом виджете	
QScrollBar	Вертикальная или горизонтальная полоса прокрутки	
QSizeGrip	Область захвата для изменения размера окна верхнего уровня	
QSlider	Вертикальный или горизонтальный ползунок (slider)	
QSpinBox	Виджет счетчика	
QTabBar	Панель вкладок для использования, например, в диалогах со вкладками	
QTabWidget	Стек виджетов со вкладками	
QTextEdit	Редактор многострочного текста	класс QTxContainer
QTimeEdit	Виджет для задания времени, основанный на виджете QDateTimeEdit	
QToolBox	Вертикальный набор элементов виджетов со вкладками	
QToolButton	Кнопка быстрого доступа к командам или настройкам, обычно используется в QToolBar	
QWidget	Базовый класс для всех объектов интерфейса пользователя	класс QTxContainer

ПРИЛОЖЕНИЕ В

Интерфейсные функции библиотеки QTXLib, их краткое описание

Таблица В.1 - Функции библиотеки TXLib

№ п/п	Прототип функции	Описание функции
1	void txInit(const char *s)	Инициализирует приложение Qt без создания окна.
2	bool txSetDefaults()	Устанавливает параметры рисования значениями по умолчанию.
3	int txGetExtentX()	Возвращает ширину окна рисования.
4	int txGetExtentY()	Возвращает высоту окна рисования.
5	bool txOK()	Проверяет корректность работы библиотеки.
6	POINT txGetExtent()	Возвращает ширину и высоту окна рисования в виде структуры.
7	void txGetWindowRect(TXWINDOW w, RECT *r)	Возвращает текущее положения окна рисования.
8	const char* txVersion()	Возвращает текущий номер версии библиотеки.
9	const char* txGetModuleFileName (bool fileNameOnly /*= true*/)	Возвращает имя модуля библиотеки.
10	TXWINDOW txCreateWindow(int width, int height, const char *title = NULL);	Создает и отображает на экране окно с заданными координатами, размером и заголовком. Возвращает указатель на созданное окно.
11	TXWINDOW txCreateWindow(int x, int y, int width, int height, const char *title = NULL)	То же
12	bool txLine(int x, int y, int x1, int y1)	Рисует линию.
13	bool txCircle(int x, int y, int r)	Рисует окружность.
14	bool txEllipse(int x, int y, int r, int r2)	Рисует эллипс.
15	bool txRectangle(int x, int y, int x1, int y1)	Рисует прямоугольник.
16	int txExec()	Запускает основной цикл обработки событий приложения.
17	bool txSetColor(COLORREF color, double thickness=1)	Устанавливает текущий цвет линий.
18	bool txSetColor(unsigned red, unsigned green, unsigned blue)	То же
19	COLORREF txGetColor()	Возвращает текущий цвет линий.

Продолжение таблицы В.1

№ п/п	Прототип функции	Описание функции
20	bool txSetFillColor(COLORREF color)	Устанавливает текущий цвет заливки.
21	bool txFillColor(unsigned red, unsigned green, unsigned blue)	То же
22	COLORREF txGetFillColor()	Возвращает текущий цвет заливки.
23	int txExtractColor (COLORREF color, COLORREF component)	Извлекает цветовую компоненту (цветовой канал) из смешанного цвета.
24	COLORREF txRGB2HSL (COLORREF rgbColor)	Преобразует цвет RGB в HSL.
25	COLORREF txHSL2RGB (COLORREF hslColor);	Преобразует цвет HSL в RGB.
26	bool txSelectFont(const char name[], int sizeY, int sizeX = -1, int bold = 0, bool italic = false, bool underline = false, bool strikeout = false, double angle = 0)	Устанавливает текущий шрифт и аттрибуты для выводимого текста.
27	unsigned txSetTextAlign (unsigned align)	Устанавливает выравнивание для выводимого текста.
28	bool txTextOut(int x, int y, const char *text)	Выводит текст на экран.
29	bool txArc(double x0,double y0, double x1, double y1, double startAngle, double totalAngle)	Рисует дугу.
30	bool txPie(double x0, double y0, double x1, double y1, double startAngle, double totalAngle)	Рисует сектор эллипса.
31	bool txChord (double x0, double y0, double x1, double y1, double startAngle, double totalAngle)	Рисует хорду эллипса.
32	bool txTriangle(double x1, double y1, double x2, double y2, double x3, double y3)	Рисует треугольник
33	bool txSetPixel(double x, double y, COLORREF color)	Рисует пиксель (точку) на экране.
34	bool txPixel(double x, double y, double red, double green, double blue)	То же.
35	bool txClear()	Очищает окно.
36	COLORREF txGetPixel(double x, double y)	Возвращает цвет пикселя (точки) с заданными координатами.
37	void txGetPixel(double x, double y, int *r, int *g, int *b)	То же.

Продолжение таблицы В.1

№ п/п	Прототип функции	Описание функции
38	bool txPolygon (const POINT points[], int numPoints)	Рисует многоугольник по заданным точкам.
39	double txSleep (double time)	Ожидает заданное количество миллисекунд.
40	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.
41	bool In (const POINT& pt, const RECT& rect	Проверяет вхождение точки в заданную область.
42	bool In (const COORD& pt, const SMALL_RECT& rect)	То же.
43	int random (int range)	Возвращает случайное число в диапазона от 0 до range.
44	double random (double left, double right)	Возвращает случайное число в заданном диапазоне.
45	bool txFloodFill (double x, double y, COLORREF color=TX_TRANSPARENT, DWORD mode=0)	Закрашивает область заданным цветом до заданной границы, начиная от заданной точки.
46	int txGetTextExtentY (const char text[])	Возвращает высоту текстовой надписи.
47	int txGetTextExtentX (const char text[])	Возвращает ширину текстовой надписи.
48	SIZE txGetTextExtent (const char text[])	Возвращает высоту и ширину текстовой надписи в виде структуры.
49	bool txDrawText (double x0, double y0, double x1, double y1, const char text[], unsigned format = 0)	Выводит текст на экран.
50	bool txIDontWantToHaveAPauseAfterMyProgramBeforeTheWindowWillClose_AndIWillNotBeAskingWhereIsMyPicture()	Ничего не делает. Оставлена для совместимости с TXLib.
51	bool txSelectObject(unsigned obj)	Выбирает заданный объект. Оставлена для совместимости с TXLib.
52	int txUpdateWindow(int update=true)	Обновляет окно рисования.
53	int txBegin(bool mode=false)	Начинает рисовать без вывода на экран. Позволяет значительно повысить быстродействие в отдельных случаях.
54	int txEnd()	Выводит на экран разом все что нарисовано после txBegin().

Продолжение таблицы В.1

№ п/п	Прототип функции	Описание функции
55	bool txLock(bool wait=true)	Блокировка окна рисования. Оставлена для совместимости с TXLib.
56	bool txUnlock()	Разблокировка окна рисования. Оставлена для совместимости с TXLib.
57	bool txDestroyWindow()	Удаляет окно рисования.
58	double txQueryPerformance()	Возвращает производительность системы. Оставлена для совместимоси с TXLib,
59	HDC txDC()	Возвращает дескриптор контекста рисования холста.
60	HDC txCreateCompatibleDC(double sizeX, double sizeY, HBITMAP bitmap=NULL, int size=0)	Создает дополнительный холст (контекст рисования, Device Context, DC) в памяти.
61	HDC txLoadImage(const char filename[], unsigned imageFlags=0, unsigned loadFlags=0);	Загружает изображение в виртуальный холст и возвращает его дескриптор.
62	bool txDeleteDC(HDC dc)	Удаляет виртуальный холст из памяти.
63	bool txBitBlt(HDC dest, double xDest, double yDest, double width, double height, HDC src, double xSrc=0, double ySrc=0, DWORD rOp=0)	Копирует изображение с одного холста (контекста рисования, DC) на другой.
64	bool txTransparentBlt(HDC dest, double xDest, double yDest, double width, double height, HDC src, double xSrc=0, double ySrc=0, COLORREF transColor=TX_BLACK)	Копирует изображение с одного холста (контекста рисования, DC) на другой с учетом прозрачности.
65	bool txAlphaBlend (HDC dest, double xDest, double yDest, double width, double height, HDC src, double xSrc=0, double ySrc=0, double alpha=1.0)	Копирует изображение с одного холста (контекста рисования, DC) на другой с учетом прозрачности.
66	void txKeyEvent(KEYEVENTHANDLER f)	Назначает пользовательский обработчик событий от клавиатуры.
67	void txMouseEvent(MOUSEEVENTHANDLER f)	Назначает пользовательский обработчик событий от мыши.
68	void txTimerEvent(TIMEREVENTHANDLER f, void *p, unsigned time)	Назначает пользовательский обработчик событий по таймеру с заданной задержкой.
69	unsigned txMessageBox(const char *text, const char *header="TXLib сообщает", unsigned flags=0);	Выводит сообщение в окне.
70	const char* txInputBox(const char *text = NULL, const char *caption = NULL, const char *input = NULL)	Запрашивает у пользователя ввод строки в окне.

Продолжение таблицы В.1

№ п/п	Прототип функции	Описание функции
71	bool txDrawDC(double x, double y, HDC dc)	Выводит вирутальный холст (изображение) на экран.
72	void txSetWindow(TXWINDOW w)	Устанавливает текущее активное окно.
73	TXWINDOW txGetWindow()	Возвращает текущее активное окно.
74	bool txNotifyIcon(unsigned flags, const char title[], const char format[])	Выводит всплывающее сообщение в системном трее.
75	bool txPlaySound(const char filename[]=NULL, DWORD mode=0)	Воспроизводит звук из файла, но пока ничего не делает.
76	QPushButton* txButton(int x, int y, int width, int height, const char *caption, TXSLOT p = NULL, char *xml = NULL)	Вставляет в окно кнопку и возвращает указатель на нее.
77	QTextEdit* txTextEdit(int x, int y, int width, int height, const char *caption, TXSLOT p = NULL, char *xml = NULL)	Вставляет в окно редактор многострочного текста с прокруткой и возвращает указатель на него.
78	QLineEdit* txEdit(int x, int y, int width, int height, const char *caption, TXSLOT p = NULL, char *xml = NULL)	Вставляет в окно строку ввода и возвращает указатель на нее.
79	QLabel* txLabel(int x, int y, int width, int height, const char *caption, TXSLOT p = NULL, char *xml = NULL)	Вставляет в окно произвольную надпись (метку) и возвращает указатель на нее.
80	QCheckBox* txCheckBox(int x, int y, int width, int height, const char *caption, bool *state, TXSLOT p = NULL, char *xml = NULL)	Вставляет в окно флажок и возвращает указатель на него.
81	QRadioButton* txRadioButton(int x, int y, int width, int height, const char *caption, bool *state, TXSLOT p = NULL, char *xml = NULL)	Вставляет в окно одиночную радиокнопку и возвращает указатель на нее.
82	QGroupBox* txRadioGroup(int x, int y, int width, int height, const char *caption, int *id, TXSLOT p, char *xml, const char *s, …)	Вставляет в окно группу радиокнопок, событийно связанных между собой и возвращает указатель на нее.
83	QGroupBox* txRadioGroup(int x, int y, int width, int height, const char *caption, int *id, TXSLOT p, char *xml, char **list)	То же
84	QComboBox* txComboBox(int x, int y, int width, int height, const char *caption, int *id, TXSLOT p, char *xml, char **list)	Вставляет в окно комбинированный список и возвращает указатель на него.

Окончание таблицы В.1

№ п/п	Прототип функции	Описание функции
85	QListWidget* txListBox(int x, int y, int width, int height, const char *caption, int *id, TXSLOT p, char *xml, char **list)	Вставляет в окно список с прокруткой и возвращает указатель на него.

ПРИЛОЖЕНИЕ Г

Примеры использования QTXLib для решения практических задач

Пример 1. Программа для демонстрации всех основных возможностей библиотеки — работы с графикой, диалогами, использования одновременно нескольких окон, обработки событий. Скриншот, демонстрирующий работу программы представлен на рисунке Г.1. Программа использует вывод в консоль для извещения пользователя о событиях элементов управления диалога.

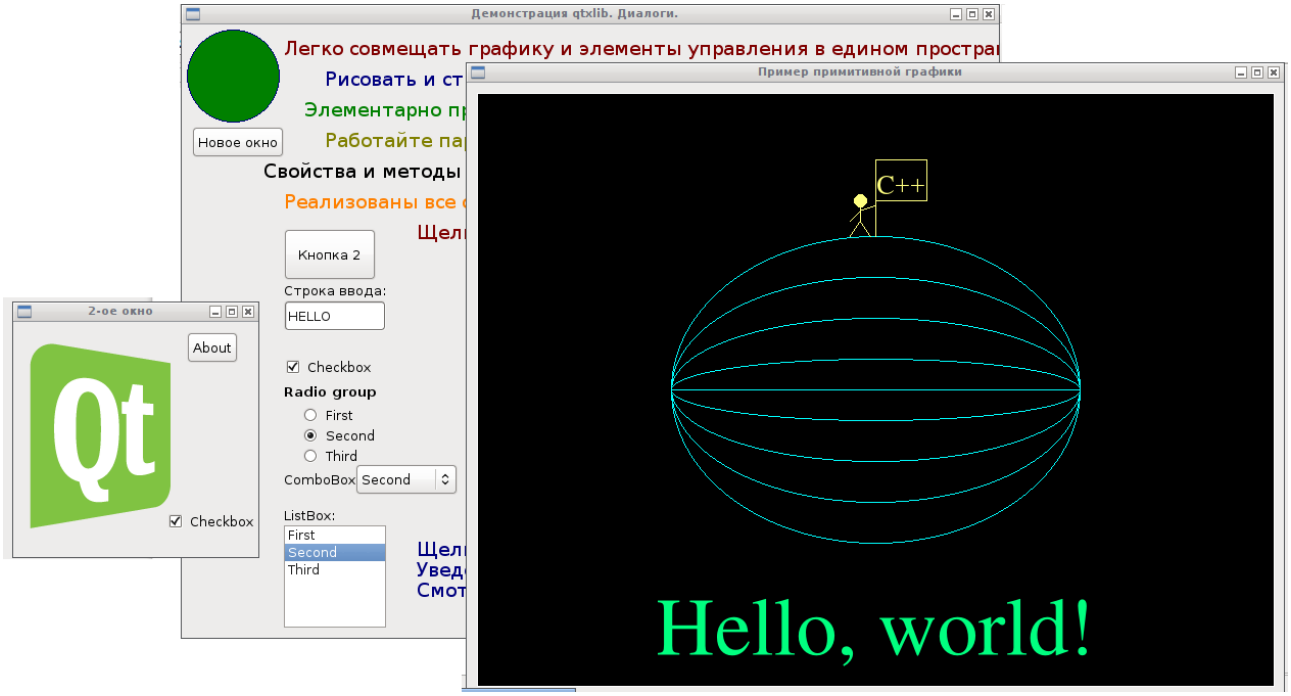

Рисунок Г.1 — Скриншот примера работы с графикой, событиями и элементами диалога

Пример 2. Простая программа «Калькулятор». Демонстрирует работу с элементами диалога, их взаимодействие между собой, возможность использования методов Qt для изменения их свойств и получения доступа к данным в этих элементах. Скриншот программы представлен на рисунке Г.2.

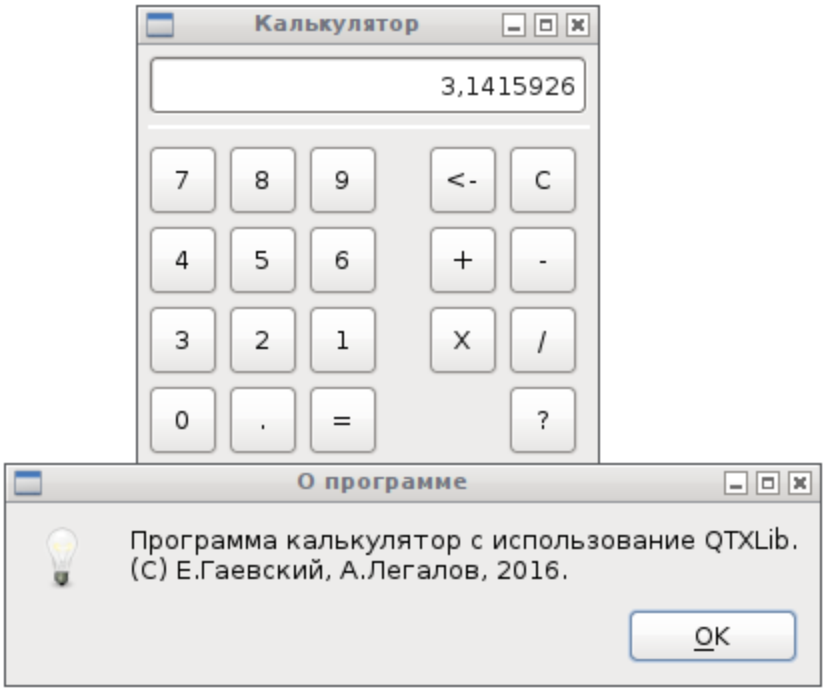

Рисунок Г.2 — Скриншот программы «Калькулятор»

Пример 3. Графическая игровая программа «Питон». Змея ползает по полю и ест кроликов. Каждый съеденный кролик увеличивает ее длину на пять единиц, однако от столкновения с препятствием змея погибает. Задача игрока — достичь максимальной длины. Управление змеей осуществляется клавишами управления курсором. Программа отражает как графические возможности библиотеки QTXLib, так и обработку событий по таймеру и от клавиатуры. Скриншот программы представлен на рисунке Г.3.

Рисунок Г.3 - Скриншот графической игровой программы «Питон»

Пример 4. Графическая игровая программа «Арканоид». По игровому полю летает мяч и сбивает кирпичи. Игроку необходимо, управляя ракеткой клавишами «влево» и «вправо», сбить все кирпичи и не упустить мяч. Программа отражает как графические возможности библиотеки QTXLib, так и обработку пользователем событий по таймеру и от клавиатуры. Скриншот программы представлен на рисунке Г.4.

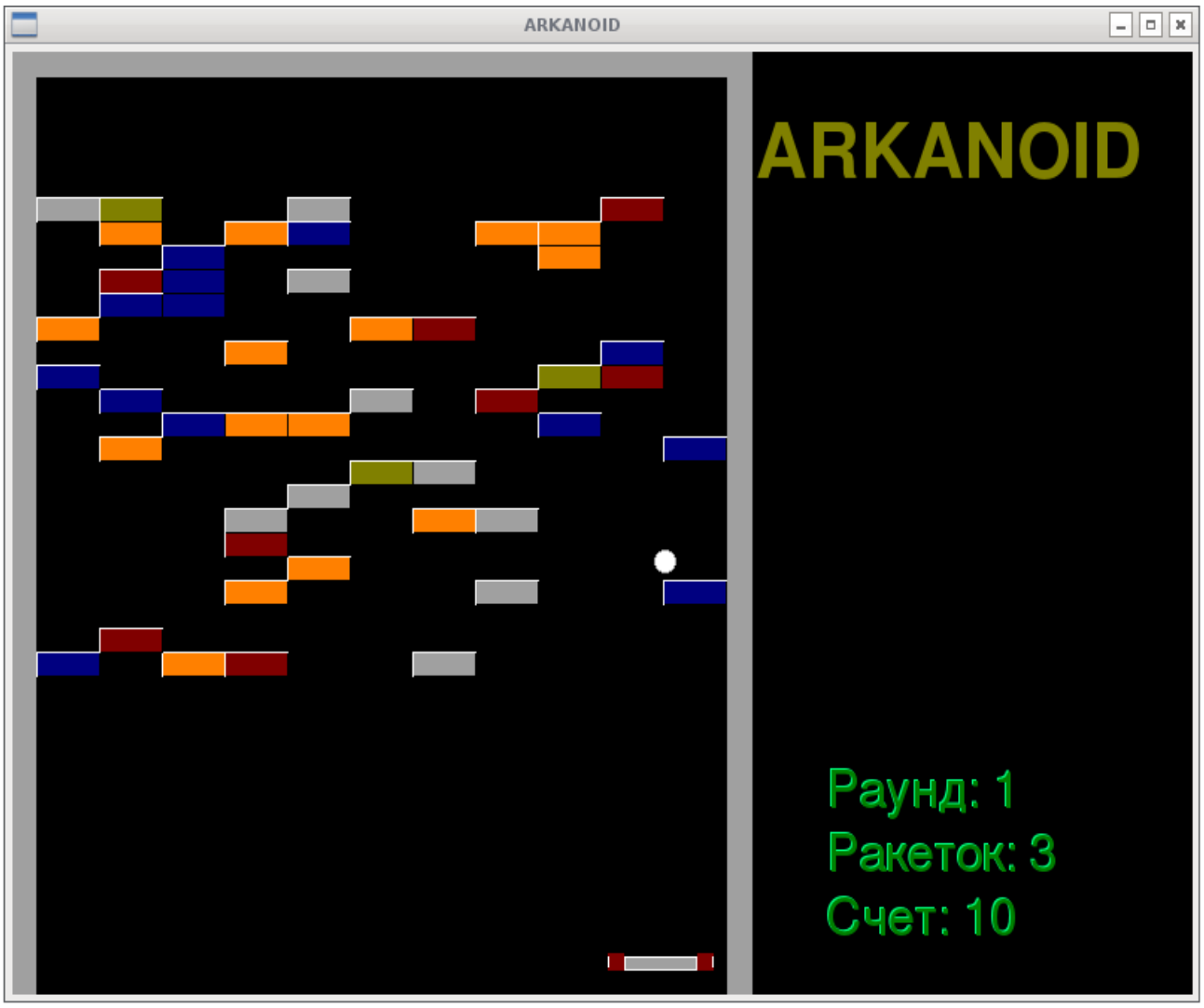

Рисунок Г.4 — Скриншот графической игровой программы «Арканоид»

ПРИЛОЖЕНИЕ Д

Переносимость библиотеки QTXLib. Демонстрация работы библиотеки на ARM микрокомпьютере Raspberry Pi 3

Демонстрация работы библиотеки на Raspberry Pi 3 Model B показана на рисунке Д.1.

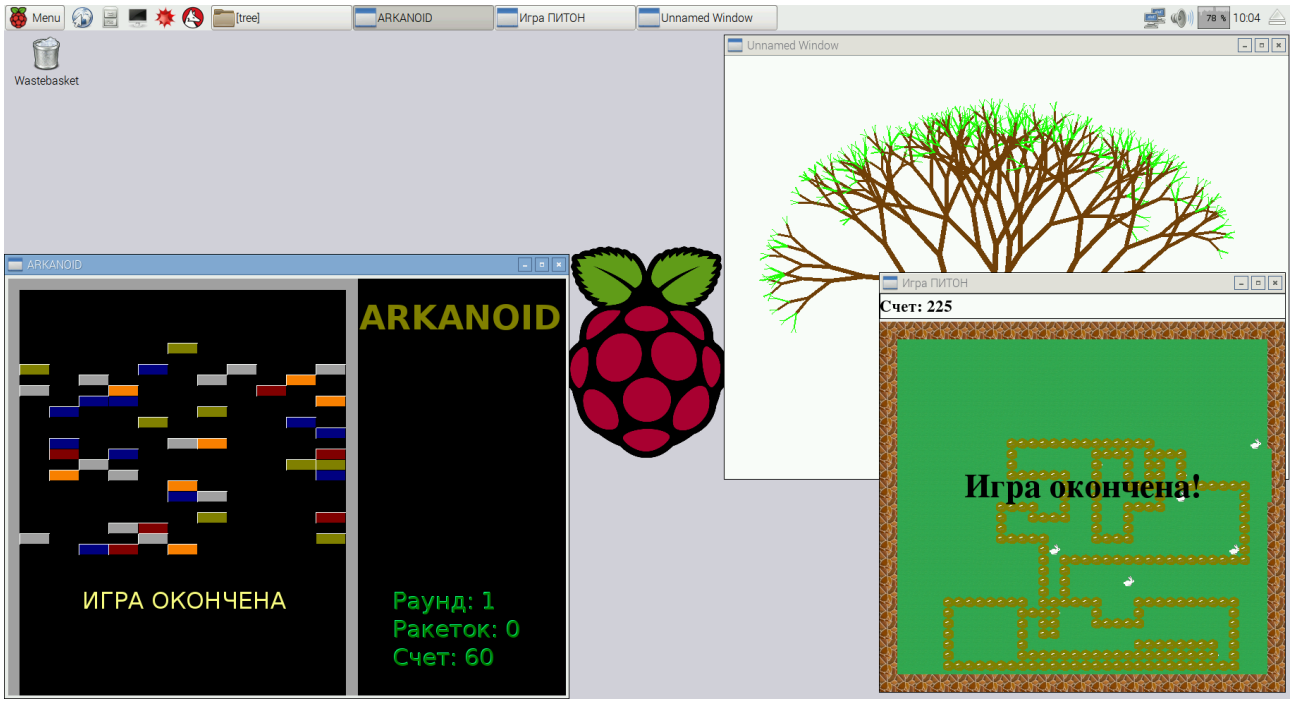

Рисунок Д.1 — Демонстрация работы библиотеки на Raspberry Pi 3 Model B