Петрова О.Б.

Разработка приложений на основе wxWidgets

Что такое wxWidgets

wxWidgets — это кроссплатформенная библиотека инструментов с открытым исходным кодом для разработки кроссплатформенных на уровне исходного кода приложений, в частности для построения графического интерфейса пользователя (GUI).

Интерфейс пользователя представляет собой совокупность средств и методов, при помощи которых пользователь взаимодействует с различными, чаще всего сложными, машинами, устройствами и аппаратурой.

Графический интерфейс пользователя - разновидность пользовательского интерфейса, в котором элементы интерфейса (меню, кнопки, значки, списки и т. п.), представленные пользователю на дисплее, исполнены в виде графических изображений.

wxWidgets разработана не только для того, чтобы создавать GUI. Она также имеет набор классов для работы с графическими изображениями, HTML, XML документами, архивами, файловыми системами, процессами, подсистемами печати, мультимедиа, сетями, классы для организации многопоточности, отладки, отправки дампов и множество других инструментов.

Особенность, которая отличает wxWidgets от многих других фреймворков, таких как MFC и OWL, — это многоплатформенная природа. wxWidgets имеет API, который одинаков или почти одинаков на всех поддерживаемых платформах. Это означает, что вы можете написать приложение, к примеру, под Windows и затем с очень немногими (если они вообще понадобятся) изменениями перекомпилировать его под Linux или Mac OS X. Это дает огромное преимущество по сравнению с раздельным программированием под каждую платформу: не придется изучать специализированный API для каждой платформы. Кроме того, при выпуске новых версий операционных систем и версий wxWidgets уже

написанные приложения всегда будут выглядеть актуально и использовать последние графические версии элементов управления. Другая отличительная особенность — wxWidgets использует графические элементы интерфейса операционной системы.

wxWidgets — бесплатная свободная платформа. Это важный момент. Создавая исходный код с использованием бесплатных свободных библиотек, всегда можно решить свои задачи, редактируя исходный код; это гораздо практичнее, чем отправлять запрос на требуемую функциональность производителю библиотеки и дожидаться обновления.

wxWidgets имеет широкий спектр пользователей — от программистов-энтузиастов до крупных корпораций, от отделов научного программирования до медицинских исследовательских групп, от экологических исследователей до телекоммуникационной индустрии.

Почему нужно использовать wxWidgets? Потому что вы хотите быстро и просто написать GUI-приложение, которое работает на разных платформах. Однако, несмотря на свои достоинства, библиотека wxWidgets, отстает по популярности от своих конкурентов Qt, MFC. Причина кроется в недостаточном количестве учебных материалов, в т.ч. на русском языке.

Сравнительная характеристика кроссплатформенных библиотек

Продукты с открытым кодом всегда противопоставляют их проприетарным эквивалентам. Используя wxWidgets, известно, что код, на который вы полагаетесь, никогда не исчезнет. Всегда можно самостоятельно исправить любую проблему, поправив исходный код. Можно получить большое удовольствие, принимая участие в обсуждении проекта, нежели общаясь с персоналом, осуществляющим корпоративную поддержку проприетарных продуктов. Когда мы используем wxWidgets, то попадаем во впечатляющее сообщество талантов, обладающих большими знаниями. Многие аспекты разработки приложений, которые в ином случае пришлось бы придумывать самостоятельно, инкапсулированы этими разработчиками в легко используемые классы, которые можно подключить к своему коду.

Активное пользовательское сообщество проконсультирует в случае необходимости в списках рассылки. В рассылке можно обсудить не только wxWidgets, но также и другие вещи, близкие сердцу любого разработчика.

Библиотека wxWidgets широко поддерживается компьютерной индустрией. Список пользователей включает в себя такие известные компании как AOL, AMD, CALTECH, Lockheed Martin, NASA, Open Source Applications Foundation, Xerox и многие другие. wxWidgets объединяет огромное число пользователей: от программистов-одиночек до огромных корпораций, от отделений компьютерных наук до центров медицинских исследований и от экологических организаций до индустрии телекоммуникаций. Библиотека используется множеством открытых проектов, таких как редактор музыки Audacity и системой управления базами данных pgAdmin III.

Люди используют wxWidgets по различным причинам: для одних это простая и элегантная замена MFC, для других — возможность легко менять платформу и перейти, например, с Microsoft Windows на Unix или Mac OS X. Проект wxWidgets также уделяет внимание поддержке мобильных платформ.

Одним из весомых преимуществ проекта Qt является наличие качественной документации, в отличие, например, от wxWidgets. Статьи документации снабжены большим количеством примеров. Исходный код самой библиотеки хорошо форматирован, подробно комментирован и легко читается, что также упрощает изучение Qt. Основным недостатком wxWidgets был недостаток документации. Так, на официальном сайте в разделе «Документация» по многим классам не было справки вообще, либо справка ограничивалась одной-двумя строками.

Возможности библиотеки wxWidgets

1. Основы построения графических изображений

Рисование в wxWidgets всегда производится на контексте устройства, который является классом-наследником от wxDC. В библиотеке нет такой вещи, как рисование непосредственно в окне. Вместо этого вы создаете

контекст устройства для окна и рисуете на этом контексте. Также существуют отдельные контексты устройств для изображений и принтеров. Приятным следствием использования такой абстракции является то, что вы можете сделать один и тот же код рисования, и он будет работать на множестве различных контекстов устройств: просто параметризуете его с помощью wxDC и, если необходимо, передайте контексту правильный коэффициент масштабирования. Опишем основные свойства контекста устройства. Контекст устройства имеет свою собственную систему координат с началом в левом верхнем углу. Позиция начала координат может быть изменена с помощью метода SetDeviceOrigin, в результате чего будет казаться, что контекст устройства сдвинут. Данный метод часто используется при рисовании на wxScrolledWindow. Также имеется возможность изменить ориентацию осей с помощью SetAxisOrientation. К примеру, можно установить направление оси у снизу наверх.

Существует различие между логическими (logical) и физическими (device) единицами измерения. Физические — это единицы измерения, специфичные для данного устройства. В частности для экрана таким единицами измерения являются пиксели. Для принтера размер его физических параметров определяется разрешением принтера и может быть получен с помощью методов GetSize (размер страницы в физических единицах) или GetSizeMM (размер страницы в миллиметрах). Режим отображения в контексте устройства определяет единицы измерения, используемые для преобразования логических единиц в физические. помощью метода SetUserScale, в который передается соответствующий коэффициент. Например, в режиме wxMM_TEXT коэффициент (1.0, 1.0) делает логические и физические единицы одинаковыми. По умолчанию используется режим отображения wxMM_TEXT и коэффициент (1.0, 1.0). Контекст устройства имеет область отсечения, координаты которой устанавливаются с помощью SetClippingRegion и очищаются с помощью DestroyClippingRegion. За границами отсечения графика не отрисовывается.

Одним из полезных использований данного свойства является отрисовка строки, которая должна быть только внутри заданного прямоугольника, даже если строка превышает его размеры. Чтобы это сделать будет достаточно установить размер области отсечения равной прямоугольнику, написать текст и уничтожить область отсечения. В результате текст будет находится только внутри заданной области. Как и у настоящих художников, в начале необходимо выбрать инструмент для рисования. Любые операции рисования используют для обводки изображения текущее перо, а для заливки — текущую кисть. Текущий шрифт, вместе с внешним (foreground) и задним (background) фоном определяют вид отображаемого текста.

2. Контексты устройств

В библиотеку включены следующие классы:

· wxClientDC. Для рисования в клиентской области окна.

· wxBufferedDC. Замена для wxClientDC с двойной буферизацией.

· wxWindowDC. Для рисования на клиентской и неклиентской (декорированной) области окна. Это очень редко используемый класс, который реализован не для всех платформ.

· wxPaintDC. Для рисования в клиентской области окна во время обработки сообщения о рисовании.

· wxBufferedPaintDC. Замена для wxPaintDC с двойной буферизацией.

· wxScreenDC. Для рисования на экране или копирования с него.

· wxMemoryDC. Для рисования на битовой карте или копирования с нее.

· wxMetafileDC. Для создания метафайла (в Windows и Mac OS X).

· wxPrinterDC. Для рисования на принтере.

· wxPostScriptDC. Для рисования в PostScript-файл или принтер.

Рисование в окне с использованием wxClientDC

Следует использовать объект wxClientDC для рисования на клиентской области окна (но только не в обработчике сообщения о рисовании). Например, для реализации эскизного приложения может понадобится создать объект wxClientDC внутри обработчика сообщений от мыши.

Данный объект также можно использовать внутри обработчика сообщения о закраске фона. Альтернативой использования wxClientDC является класс wxBufferedDC, который хранит результат рисования в контексте устройства памяти и посылает изображение в окно за один раз при уничтожении данного контекста. В результате обновления происходят более гладко и подходят для случая, чтобы пользователь не видел попиксельное обновление экрана. Использование данного класса почти ничем не отличается от использования wxClientDC. В целях эффективности можно передавать сохраненную битовую карту в конструктор класса, чтобы избежать ее создание каждый раз.

3. Рисование в окне с помощью wxPaintDC

Если переопределять обработчик события о рисовании, то всегда необходимо создавать внутри объект wxPaintDC, даже если он не нужен. Создание этого объекта говорит wxWidgets, что поврежденная область окна уже перерисована, а поэтому оконная система не должна посылать сообщение о рисовании до бесконечности. В сообщении о рисовании следует вызвать метод wxWindow::GetUpdateRegion(), чтобы получить поврежденную область или wxWindow::IsExposed(), чтобы определить находится ли данная точка или прямоугольник в поврежденном регионе. Если это возможно, то просто перерисовать данный регион. Контекст устройства автоматически передает необходимую область отсечения внутри события о рисовании, но можно ускорить работу перерисовав только то, что необходимо. Событие о перерисовке генерируется в случае, когда действия пользователя приводят к тому, что окну необходимо перерисовать часть своей области, но это событие также может быть следствием вызова метода wxWindow::Refresh() или wxWindow::RefreshRect(). Если точно известно какую область необходимо перерисовать, то можно обновить только эту область, чем максимально снизить возможное мерцание. Главной проблемой при обновлении окна является то, что невозможно гарантировать момент, когда оно на самом деле будет обновлено. Если необходимо форсировать

обновление окна (например во время продолжительных вычислений), то можно вызвать метод wxWindow::Update после предварительного вызова Refresh или RefreshRect.

4. Рисование на битовых картах с помощью wxMemoryDC

Контекст устройства в памяти всегда имеет связанную с ним битовую карту, поэтому рисование в такой контекст означает рисование на связанной битовой карте. Контекст в памяти использовать достаточно просто. Сначала с помощью конструктора по умолчанию создается объект класса wxMemoryDC, далее вызывается метод SelectObject для привязки битовой карты к контексту. После использования контекста памяти необходимо вызвать SelectObject с параметром wxNullBitmap, чтобы удалить связывание. Также можно скопировать область из контекста в памяти в другой контекст с помощью метода Blit.

5. Использование wxMetafileDC для создания метафайлов

Класс wxMetafileDC доступен на платформах Windows и Mac OS X, где он служит прослойкой для создания метафайлов Windows и Mac PICT соответственно. Данный класс предоставляет возможность рисования в объект wxMetafile, который содержит список инструкций для слоя рисования, которые могут интерпретироваться самим приложением или отрисоваться в контексте устройства с помощью wxMetafile::Play().

6. Рисование на экране при помощи wxScreenDC

Класс wxScreenDC используется, чтобы рисовать в произвольной области экрана.

Это иногда полезно, если необходимо предусмотреть некоторую визуальную реакцию на операцию перетаскивания. В целях оптимизации можно ограничить область на экране, определив регион (чаще всего равным размерам самого окна). Кроме того данный класс можно использовать, чтобы захватить изображение с экрана (сделать снимок).

7. Печать с использованием wxPrinterDC и wxPostScriptDC

wxPrinterDC является поверхностью для печати. В системах Windows и Mac

этот класс напрямую взаимодействует с соответствующей подсистемой печати. В Unix-системах, где отсутствует стандартная модель печати, обычно используется wxPostScriptDC. Если в системе доступна система печати GNOME, то лучше использовать именно ее.

Существует несколько способов создания объекта wxPrinterDC. Например, можно передать объект wxPrintData, в котором предварительно можно установить тип бумаги, ориентацию, число копий и так далее. Самый простой путь — показать wxPrintDialog, а потом вызвать wxPrintDialog::GetPrintDC() для получения подходящего контекста wxPrinterDC с установками, выбранными пользователем. На более высоком уровне можно создать наследника от wxPrintout и определить в нем поведение при печати и при предпросмотре, а когда необходимо передать класс экземпляру объекта wxPrinter.

Если необходимо печатать по большей части только текст, то лучше использование класса wxHtmlEasyPrinting, который позволит избежать использования wxPrinterDC или wxPrintout: просто необходимо создать корректный HTML-файл (используя поддерживаемое wxWidgets подмножество синтаксиса HTML) и создать объект wxHtmlEasyPrinting для его печати или предпросмотра.

Класс wxPostScriptDC — это контекст устройства, предназначенный для создания PostScript-файлов, которые можно посылать непосредственно принтеру. Хотя данный класс и создан по большей части для Unix-систем, но его также можно использовать и для других систем, когда необходимо создать файл в формате PostScript, но у вас нет гарантии, что установлен PostScript-драйвер. Можно создать wxPostScriptDC, передав ему объект wxPrintData или имя файла для сохранения. Дополнительный булевый параметр говорит о том, показывать ли диалог печати пользователю.

8. Файловые системы.

wxWidgets содержит множество различных функций и классов для платформонезависимой работы с файлами. wxFile может быть использован для низкоуровневого ввода/вывода. Он содержит все обычные функции для работы с целочисленными файловыми дескрипторами (открытие/закрытие, чтение/запись, позицирование и т.п.), но в отличие от стандартных функций языка С данный класс сообщает об ошибках через wxLog и автоматически закрывает файл в деструкторе. Можно создать объект класса wxFile, используя конструктор по умолчанию, а далее вызвать метод Create или Open. Также можно использовать конструктор, который в качестве параметров принимает имя открываемого файла и режим работы:

- wxFile::read() — для чтения,
- wxFile::write() — для записи,
- wxFile::read_write() — для чтения и записи.

Вы также можете создать wxFile из существующего файлового дескриптора, который можно передать в конструктор или с помощью метода Attach. wxFile можно принудительно закрыть, вызвав метод Close(), который также вызывается автоматически при уничтожении объекта.

Для чтения данных из файла существует метод Read(), которому передается указатель на буфер void* и число байт для чтения. Функция возвращает число считанных байтов или wxInvalidOffset, в случае ошибки. Следует использовать метод Write(), чтобы записать буфер void* или строку типа wxString в файл. Метод Flush() вызывает немедленную запись измененных данных в файл.

Для проверки достижение конца файла следует использовать метод Eof, который возвращает true, если указатель находится к конце файла (это отличается от поведения функции Eof класса wxFFile, которая возвращает true только после попытки прочитать информацию, расположенную за последним байтом). Размер файла можно определить вызвав Length().

Seek и SeekEnd устанавливают текущую позицию в файле, отсчитывая смещение от начала или конца файла соответственно. Tell возвращает

текущее смещение от начала файла с типом wxFileOffset (64-битовое целое число в случае поддержки платформой или 32-битовое целое число в противном случае).

Вызвав статическую функцию Access() можно определить возможно ли открыть файл в выбранном режиме. Другая статическая функция Exists() проверяет существование указанного файла.

wxTextFile реализует довольно прямолинейный путь для чтения и записи ма-
леньких текстовых файлов и их построчной обработки.
Метод Open необходимо использовать, чтобы прочитать файл в память и разбить его на строки. Метод Write позволяет сохранить информацию в файл. GetLine или прямое обращение по индексу в массиве позволяет получить требуемую строку. Также перебор строк можно осуществить, используя GetFirstLine, GetNextLine и GetPrevLine.

Функции AddLine и InsertLine позволяют добавить строку в конец файла, а удалить требуемую строку можно с помощью RemoveLine. Для полной очистки содержимого файла существует метод Clear.

wxTempFile реализует относительно безопасный метод для замены содержимого существующего файла. Класс wxTempFile является потомком от wxFile и использует временные файлы для записи данных. При этом реальная перезапись выбранного файла происходит только после вызова метода Commit. Использование временных файлов для записи пользовательских данных является отличной идеей, так как в этом случае минимизируются последствия возможных ошибок, которые могут произойти во время записи (таких как выключение электричества, ошибки в программе или других катаклизмов). При использовании временных файлов такие неприятности не смогут повредить текущему файлу на диске.

wxDir — это переносимый эквивалент функций open/read/closedir из системы

unix, которые поддерживают перечисление файлов в каталоге. wxDir поддержива-
ет перечисление как файлов, так и каталогов. Данный класс предоставляет гибкий
механизм для рекурсивного перечисления файлов, а также более простую в использовании функцию GetAllFiles. После открытия каталога с помощью Open (или передав нужный путь в конструктор) следует вызвать метод GetFirst, передав ему в качестве параметра указатель на строку в которую будет записано найденное имя файла. Дополнительно можно передать фильтр для файлов (по умолчанию пустая строка означает отсутствие фильтра) и необязательные флаги. Далее вызов GetNext до тех пор пока функция не возвратит false. Фильтр может содержать групповые символы, такие как «*» (означает любое число любых символов) и «?» (означает один любой символ). Поле с флагами может быть комбинацией одной или нескольких битовых констант wxDIR_FILES (искать файлы), wxDIR_DIRS (искать каталоги), wxDIR_HIDDEN (искать скрытые файлы) и wxDIR_DOTDOT (добавлять к результату псевдокаталоги «..» и «...»). По умолчанию ищутся все возможные объекты, кроме «..» и «...».

Классы потоков

Потоки — это высокоуровневая модель для чтения и записи данных. Можно писать свой код не заботясь о том куда осуществлять запись: в файлы, в память или даже в сокеты. Некоторые классы wxWidgets (например, wxImage), поддерживающие чтение/запись файлов, также могут использовать ввод/вывод в поток.

wxStreamBase является базовым классом для всех потоков и объявляет функции такие как OnSysRead и OnSysWrite, которые должны быть реализованы в дочерних классах. Его дочерние классы wxInputStream и wxOutputStream - фундамент для всех других классов, осуществляющих чтение и запись, например для wxFileInputStream и wxFileOutputStream соответственно.

Классы wxFileInputStream и wxFileOutputStream базируются на wxFile и инициализированы именем файла, объектом wxFile или целочисленным файловым дескриптором.

wxMemoryInputStream и wxMemoryOutputStream используют внутренний буфер для организации потока. Конструкторы обоих этих классов берут в качестве параметров буфер char* и его размер. Данные параметры могут быть опущены, чтобы позволить классу выделять память автоматически по мере необходимости. wxStringInputStream берет в качестве параметра строку из которой будет осуществляться чтение. wxStringOutputStream получает необязательный указатель настроку класса wxString, в которую будет осуществляться запись. Если данный параметр опущен, то класс самостоятельно создаст строку, доступ к которой можно получить через метод GetString.

9. Потоки фильтрации

wxFilterInputStream и wxFilterOutputStream являются базовыми для потоков, которые могут быть помещены над остальными потоками. Примером фильтрующего потока является поток wxZlibInputStream. Если передать ему входной поток, созданный из сжатого zlib или glib файла, то можно читать данные из потока wxZlibInputStream без необходимости заботится об их декомпрессии. Аналогично, можно создать wxZlibOutputStream и связать с ним выходной поток, такой как wxFileOutputStream. Запись данных в поток wxZlibOutputStream приведет к тому, что они будут сжаты и посланы в wxFileOutputStream.

10. Потоки со сжатием

wxZipInputStream является достаточно сложным классом, позволяющим работать с архивами, а не только с линейным потоком данных. Обычно работу с архивами производят с помощью группы классов, включающих в себя wxArchiveClassFactory и wxArchiveEntry, но существует возможность читать и писать zip-файлы без прямого их использования. Для использования wxZipInputStream можно создать поток из wxInputStream (который

самостоятельно откроет архив) или указав имя архива и имя файла внутри архива.

В wxWidgets реализован механизм виртуальных файловых систем, который позволяет читать данные из различных источников, как из обычных файлов. С помощью данного механизма можно читать из zip-архивов, из памяти и по протоколам HTTP и FTP. Хотя он не может быть использован для записи редактируемых документов (так как это инструмент для доступа только для чтения), но его можно, к примеру, использовать для доступа к ресурсам в zip-архиве. Класс wxHtmlWindow (механизм помощи wxWidgets на базе HTML) и ресурсы XRC умеют работать с виртуальными файловыми системами. Виртуальные файловые системы более удобны для работы с архивами, чем рассмотренные ранее классы, но последние позволяют не только читать, но и писать в архивы. Хотя оба механизма и используют потоки, но они никак не связаны друг с другом. В библиотеке реализованы различные виртуальные файловые системы с помощью классов-наследников от wxFileSystemHandler. Экземпляры этих объектов должны быть добавлены через wxFileSystem::AddHandler() (чаще всего это делают в функции инициализации приложения OnInit()) перед использованием этих файловых систем. Обычно все взаимодействие с файловой системой осуществляется через wxFileSystem, но иногда обработчики предоставляют функции, которые могут использоваться приложением непосредственно, как например функции AddFile() и RemoveFile() класса wxMemoryFSHandler.

11. Использование сокетов

Используя класс wxSocket из библиотеки wxWidgets, можно пересылать любые объемы данных с одного компьютера на другой. Хотя архитектура и функции сокетов очень похожи в Windows, Linux и Mac OS X, но каждая реализация API сокетов имеет свои особенности. Что более важно, события от сокетов имеют очень различные API на каждой из платформ, что часто создает сложности при их программировании. wxWidgets предоставляет удобные классы, которые позволяют легко использовать сокеты в

современных приложениях без необходимости заботится о нюансах каждой платформы в отдельности.

Ядром системы является класс wxSocketBase, который реализует основные функции для отправки и получения данных, закрытия сокета, оповещения об ошибках и так далее. Создание слушающего сокета или соединения к серверу потребует использования классов wxSocketServer или wxSocketClient соответственно. Класс wxSocketEvent используется для оповещения приложения

о событиях, происходящих с сокетом. Абстрактный класс wxSocketBase и его потомки (такие как wxIPV4address) позволяют указать имя хоста и порт. Наконец, классы потоков, такие как wxSocketInputStream и wxSocketOutputStream, можно использовать в сочетании с другими классами потоков для перемещения и преобразования данных через сокет.

12. Адресация сокета

Все классы, представляющие собой адрес сокета, являются производными от базового класса wxSockAddress, благодаря которому в методы можно легко передать адрес, независимо от используемого протокола. Класс wxIPV4address предоставляет все необходимые методы для задания удаленного хоста, используя текущую стандартную для Интернет схему адресации IPv4.

Примечание: При представлении адреса в виде беззнакового целого числа

необходимо учитывать порядок байтов в сети. Порядок байтов в сети соответствует порядку big endian (32- и 64-битные архитектуры Intel или AMD используют little endian, а архитектура Apple — big endian). В зависимости от того, как беззнаковые целые адреса хранятся или вводятся, возможно, потребуется использовать макрос wxINT32_SWAP_ON_LE, который поменяет порядок байтов на правильный только для платформы little endian. Метод Hostname может принимать строку адреса в формате wxString (например, www.wxwidgets.org) или IP-адрес в виде 4-байтового

целого числа (как уже отмечалось ранее в big endian). Без каких-либо параметров, Hostname возвращает
имя хоста в настоящий момент. Метод Service устанавливает удаленный порт, который можно задать используя строковое (wxString) описание для указания известного порта или unsigned short для указания произвольного порта. Вызов метода без параметров возвращает номер порта, выбранный в настоящий момент. IPAddress возвращает адрес удаленного хоста в виде IP-адреса в формате wxString. AnyAddress устанавливает адрес равным любому адресу текущей машины. Эквивалентен использованию адреса INADDR_ANY.

13. Клиенты сокета

Класс wxSocketClient является производным от wxSocketBase и наследует все методы базового класса. Добавленные методы предназначены для инициализации и установки подключения к удаленному серверу. Метод Connect принимает аргумент класса wxSockAddress, указывающий клиенту адрес и порт сервера для подключения. Как упоминалось ранее, необходимо использовать классы наподобие wxIPV4address, а не wxSockAddress напрямую. Второй параметр, по умолчанию равный true, указывает на то, что вызов Connect должен быть блокирующим. Если такой вызов делать в основном потоке, то на время подключения GUI будет заблокирован. Метод WaitOnConnect можно использовать после вызова Connect в случаях, если вызов Connect был деблокирующим. В первом параметре передается время ожидания в секундах, а во втором — время ожидания, но в миллисекундах Если соединение успешно установлено или запрос провалился (например, было задано неправильное имя хоста), то возвращается true. Если время ожидания истекло, то возвращается false. Если параметр задать равным -1, то будет использовано значение по умолчанию, которое составляет 10 минут, но может быть переопределено с помощью SetTimeout.

14. События от сокетов

Все события от сокетов можно отфильтровать с помощью одного макроса —EVT_SOCKET.

EVT_SOCKET(идентификатор, функция) посылает события от сокета с определенным идентификатором указанной функции. Функция должна принимать wxSocketEvent в качестве аргумента. Хотя класс wxSocketEvent сам по себе очень простой, но содержит тип события и сокет, для которого данное событие имело место. В результате отпадает необходимость хранить указатель на сокет.

15. Основные методы wxSocketEvent

wxSocketEvent используется в качестве параметра для обработчиков событий сокетов. Метод GetSocket возвращает указатель на wxSocketBase, который является сокетом, породившим данное событие. GetSocketEvent возвращает тип произошедшего события.

16. Прием и передача данных через сокет

wxSocketBase поддерживает различные базовые и продвинутые методы для чтения и записи данных через сокет. Все операции чтения и записи сохраняют результаты операции и позволяют через LastCount получить число прочитанных байт, а также последнюю ошибку, используя LastError.

17. Чтение

Discard удаляет все пришедшие данные из буфера сокета. Peek позволяет получить доступ к данным внутри буфера, не удаляя их оттуда. Методу необходимо передать буфер для считывания данных, а также его размер. Read извлекает данные из буфера сокета и копирует их в переданный буфер, вплоть до указанного размера. ReadMsg читает данные, посылаемые WriteMsg в указанный буфер с определенным размером. Если буфер переполнится, то лишние данные будут отброшены. ReadMsg всегда пытается получить все сообщение, посланное с использованием WriteMsg, если не происходит ошибка. Unread копирует данные из предоставленного буфера обратно в буфер сокета. Также необходимо указать количество данных для возвращения.

18. Запись

Write передает данные через сокет. Необходимо указать передаваемые данные и количество байт для отправки. WriteMsg очень похож на Write, но при передаче также добавляет специальный заголовок. Заголовок используется для того, чтобы клиент точно знал, сколько данных необходимо прочитать. Данные, передаваемые с использованием WriteMsg, на удаленной стороне необходимо читать с помощью ReadMsg.

19. Создание сервера

Класс wxSocketServer добавляет к wxSocketBase несколько методов для создания слушателя и приема соединений. Чтобы создать сервер необходимо определить порт для приема входящих соединений. При конструировании wxSocketServer использует тот же класс wxIPV4address, что и для wxSocketClient, но без указания удаленного хоста. В большинстве случаев, после создания серверного сокета необходимо вызвать метод Ok, чтобы проверить, что сокет успешно создан и принимает входящие соединения.

20. Основные методы wxSocketServer

wxSocketServer принимает адрес объекта с указанием порта для прослушивания, а также необязательных флагов Accept принимает входящий запрос на соединение, если таковой имеется, и создает для него сокет. Опционально можно указать необходимость ожидать новое подключение или сразу возвращать NULL, если ожидающего входящего соединения нет. Если указать необходимость ожидания, то GUI на это время будет заблокирован. AcceptWith работает аналогично Accept, но ему необходимо передать ссылку на уже существующий объект класса wxSocketBase. Возвращается логическая переменная, указывающая было ли соединение принято. WaitForAccept принимает на вход количество секунд и миллисекунд, означающее время в течении которого необходимо ожидать нового соединения. Метод возвращает true, если в указанный период времени был запрос на новое соединение и false в противном случае.

21. Обработка установления нового подключения

Когда слушающий сокет обнаруживает новое входящее соединение, то он сообщает об этом обработчику событий. Обработчик события может принять соединение и выполнить любые необходимые действия. Предполагая, что соединение будет некоторое время использоваться и сразу не закроется, можно назначить обработчик событий для нового сокета. В течение жизни программы-сервера слушающий сокет может породить тысячи новых сокетов.

Используя потоки ввода/вывода в wxWidgets, можно легко перемещать и преобразовывать большое количество данных, написав всего несколько строчек кода. Рассмотрим задачу отправки некоторого файла через сокет. Один из вариантов ее решения — открыть файл, прочитать содержимое файла в память, а затем отправить блок памяти в сокет. Такой подход хорошо работает для небольших файлов, но чтение мегабайтных или гигабайтных файлов в память будет очень долгим на компьютерах с небольшим количеством памяти, а в большинстве случаев — просто невозможным. Кроме того, что если вы хотите сжать файл перед отправкой, чтобы уменьшить сетевой трафик? Чтение больших файлов в память, последующее сжатие и запись в сокет будет просто не эффективным и не практичным.

Второй подход подразумевает чтение файла небольшими кусками (по несколько килобайт), сжатие этих кусочков и их посылку через сокет. К сожалению, сжатие отдельных кусочков будет не таким эффективным, как сжатие всего файла. Однако можно воспользоваться непрерывным сжатием (когда для сжатия одной части используется информация из предыдущей, что помогает избежать необходимости иметь свой собственный заголовок для каждой части), что приведет к необходимости написать десятки строк кода для синхронизации чтения из файла, сжатия и передачи. Библиотека wxWidgets предоставляет лучший путь решения данной проблемы.

Так как в wxWidgets реализованы классы wxSocketInputStream и wxSocketOutputStream, то появляется возможность очень легко пропустить данные из других потоков через сокеты. Учитывая, что wxWidgets также

содержит потоки для файлов, строк, текста, памяти и сжатия, то это дает возможность использования сокетов в уникальных и эффективных решениях. Вернувшись к проблеме передачи файла со сжатием, становится очевидным новое решение. Чтобы отправить файл необходимо файловый поток перенаправить потоку сжатия, а далее в сокет. На принимающей стороне мы посылаем поток данных от сокета в поток декомпрессии Zlib и, наконец, в выходной файл. Все это может быть сделано с помощью нескольких строк кода. Если делать все операции в отдельном потоке выполнения, то можно не беспокоится о блокировке GUI или нагрузке процессора в 100%, как могло произойти в первых двух решениях при передаче крупных файлов.

Возможности MFC

Пакет Microsoft Foundation Classes (MFC) даёт возможность разрабатывать GUI-приложения для Microsoft Windows на языке C++ с использованием богатого набора библиотечных классов. Большая часть MFC представляет собой относительно тонкий объектно-ориентированный слой над Windows API. Это решение, с одной стороны, повышает производительность, но, с другой стороны, наследует все недостатки дизайна Windows API и препятствует переносу программ на другие платформы.

MFC является альтернативой системам визуального программирования, таким как Delphi или Visual Basic, предназначенной для опытных программистов. Подавляющее большинство программ разрабатывается при помощи Microsoft Visual C++ и MFC. MFC - это стандарт программирования под Windows и "интернациональный язык общения". Такая ситуация объясняется многими причинами. В частности, только MFC позволяет создавать наиболее эффективные и устойчивые приложения, которые будут корректно вести себя не только в системе разработчика, но и в системах реальных пользователей. Также очень важно, что MFC поддерживает все современные технологии, реализованные в Windows, и при дополнении Windows почти сразу же дополняется и MFC.

MFC - это инструмент для программирования сложных приложений, от которых требуется высокая эффективность и надежность. MFC поощряет использование объектно-ориентированного программирования, что дает ощутимые преимущества при решении сложных (не с точки зрения только интерфейса пользователя) задач, по сравнению с компонентно-ориентированным подходом, применяемым в системах RAD (быстрой разработки приложений). Разрабатывая приложение в системе RAD, программист часто вообще не использует ООП, по крайней мере в явном виде, до тех пор, пока не соберется разработать собственный компонент. Это негативно сказывается на возможности последующего расширения возможностей. Тем не менее, не стоит воспринимать сказанное как критику систем RAD. Есть много классов приложений (например, базы данных), которые разумнее всего разрабатывать именно при помощи систем RAD, что и делают даже опытные Windows-программисты.

Библиотека содержит многоуровневую иерархию классов, насчитывающую около 200 членов. Они дают возможность создавать Windows-приложения на базе объектно-ориентированного подхода. С точки зрения программиста, MFC представляет собой каркас, на основе которого можно писать программы для Windows.

Библиотека MFC разрабатывалась для упрощения задач, стоящих перед программистом. Как известно, традиционный метод программирования под Windows требует написания достаточно длинных и сложных программ, имеющих ряд специфических особенностей. В частности, для создания только каркаса программы таким методом понадобится около 75 строк кода. По мере же увеличения сложности программы ее код может достигать поистине невероятных размеров. Однако та же самая программа, написанная с использованием MFC, будет примерно в три раза меньше, поскольку большинство частных деталей скрыто от программиста.

Одним из основных преимуществ работы с MFC является возможность многократного использования одного и того же кода. Так как библиотека

содержит много элементов, общих для всех Windows-приложений, нет необходимости каждый раз писать их заново. Вместо этого их можно просто наследовать (говоря языком объектно-ориентированного программирования). Кроме того, интерфейс, обеспечиваемый библиотекой, практически независим от конкретных деталей, его реализующих. Поэтому программы, написанные на основе MFC, могут быть легко адаптированы к новым версиям Windows (в отличие от большинства программ, написанных обычными методами).

Еще одним существенным преимуществом MFC является упрощение взаимодействия с прикладным программным интерфейсом (API) Windows. Любое приложение взаимодействует с Windows через API, который содержит несколько сот функций. Внушительный размер API затрудняет попытки понять и изучить его целиком. Зачастую даже сложно проследить, как отдельные части API связанны друг с другом! Но поскольку библиотека MFC объединяет (путем инкапсуляции) функции API в логически организованное множество классов, интерфейсом становится значительно легче управлять.

Сейчас Microsoft объявила MFC устаревшей технологией, отдавая предпочтение Windows Forms, входящим в состав .NET.

Возможности Qt

Trolltech Qt (произносится «кьют») — кросс-платформенный инструментарий разработки ПО на языке программирования C++. Есть также «привязки» ко многим другим языкам программирования: Python — PyQt, Ruby — QtRuby, Java — Qt Jambi и другие.

Qt является полностью объектно-ориентированным, легко расширяемым и поддерживающим технику компонентного программирования. 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.

Позволяет запускать написанное с его помощью ПО в большинстве современных десктопных операционных систем путём простой компиляции программы для каждой ОС без изменения исходного кода. Имеются версии для Windows, Linux, Mac OS X. До недавнего времени библиотека Qt также распространялась ещё в одной версии: Qt/Embedded. Теперь эта платформа переименована в Qtopia Core и распространяется как отдельный продукт. Qtopia Core обеспечивает базовую функциональность для всей линейки платформ, предназначенных для разработки приложений для мобильных устройств (КПК, смартфонов и т. п.). В отличие от требовательной к объему памяти систему X Windowing System, Qt/Embedded опирается на так называемые буферы кадров. Это упрощает процедуру создания быстрых и компактных приложений для устройств, работающих на платформе Linux (к ним относятся карманные ПК, мобильные телефоны и некоторые виды Internet-приставок). Главное достоинство Qt/Embedded в том, что различия в разработке ПО для встроенных систем, Windows или Linux становятся практически невидимы для программиста. Написать при помощи средств Qt несложное приложение, которое одинаково хорошо будет работать в среде Linux, Windows или на карманных устройствах, не составляет никакого труда.

Корпорация Compaq выпустила специальную версию своего карманного компьютера iPAQ, которая работает под управлением встроенного варианта Linux, созданного компанией LISA Systems, и ПО Trolltech Qt Palmtop Environment. По словам представителей LISA, встроенная система Linux обладает более эффективными по сравнению с Windows CE средствами управления энергопотреблением. А это очень важно для Compaq, поскольку iPAQ высасывает энергию из батарей, словно лимонад из бутылки.

В перспективе расширение рынка карманных устройств приведет к сокращению продаж ПК, а Qt имеет все шансы стать самой популярной платформой для создания приложений.

Qt 4 распространяется в четырех редакциях:

Qt Console — для разработки приложений без графического интерфейса (сетевые демоны, консольные приложения и т. п.).
Qt Desktop Light — облегчённая версия для разработки приложений с графическим интерфейсом, но без поддержки баз данных, сети и XML.
Qt Desktop — полная версия, включает все компоненты.
Qt Open Source Edition — полная версия для разработки свободного программного обеспечения, распространяемого под лицензией GNU GPL. Не поддерживает коммерческие компиляторы.
До версии 4.0.0 под свободной лицензией распространялись лишь Qt/Mac, Qt/X11, Qt/Embedded, но, начиная с 4.0.0 (выпущенной в конце июня 2005), Trolltech «освободили» и Qt/Windows. Следует отметить, что существовали сторонние свободные версии Qt/Windows < 4.0.0, сделанные на основе Qt/X11.

Со времени своего появления в 1996 году коммерческая версия библитеки Qt легла в основу тысяч успешных проектов во всем мире. Кроме того, Qt является фундаментом популярной рабочей среды KDE, входящей в состав многих дистрибутивов GNU/Linux.

Отличительная особенность Qt от других библиотек — использование Meta Object Compiler — предварительной системы обработки исходного кода (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 создает дополнительный исходный файл на C++, содержащий мета-объектный код.

Qt комплектуется визуальной средой разработки графического интерфейса «Qt Designer», позволяющей создавать диалоги и формы «мышью» (в режиме WYSIWYG). В поставке Qt есть «Qt Linguist» —

мощная графическая утилита, позволяющая упростить локализацию и перевод вашей программы на многие языки; и «Qt Assistant» — справочная система Qt, упрощающая работу с документацией по библиотеке, а также позволяющая создавать кроссплатформенную справку для разрабатываемого на основе Qt ПО.

Библиотека разделена на несколько модулей, для четвертой версии библиотеки это:

QtCore — классы ядра библиотеки, используемые другими модулями;

QtGui — компоненты графического интерфейса;

QtNetwork — набор классов для сетевого программирования. Поддержка различных высокоуровневых протоколов может меняться от версии к версии. В версии 4.2.x присутствуют классы для работы с протоколами FTP и HTTP. Для работы с протоколами TCP/IP и UDP предназначены такие классы как: QTcpServer, QTcpSocket для TPC/IP и QUdpSocket для UDP;

QtOpenGL — набор классов для работы с OpenGL;

QtSql — набор классов для работы с базами данных, используя язык структурированных запросов SQL. Основные классы данного модуля в версии 4.2.x: QSqlDatabase — класс для предоставления соединения с базой, для работы с какой-нибуть конкретной базой данных, требует объект унаследованный от класса QSqlDriver — абстрактного класса, который реализуется для конкретной базы данных и может требовать для компиляции SDK базы данных. Например, для сборки драйвера под базу данных FireBird/InterBase требует .h файлы и библиотеки статической линковки входящие в комплект поставки данной БД;

QtScript — классы для работы с Qt Scripts;

QtSvg — классы для отображения и работы с Scalable Vector Graphics(SVG) данными;

QtXml — модуль для работы с XML, поддерживается SAX и DOM модели работы;

QtDesigner — классы создания расширений QtDesigner'а для своих собственных виджетов;

QtUiTools — классы для обработки в приложении форм Qt Designer;

QtAssistant — справочная система;

Qt3Support — модуль с классами необходимыми для совместимости с библиотекой Qt версии 3.x.x;

QtTest — модуль для работы с UNIT тестами;

ActiveQt — модуль для работы с ActiveX и COM технологиями для Qt-разработчиков под Windows. Модуль доступен только в коммерческой редакции Qt.

Библиотека использует собственный формат проекта, именуемый .pro файлом, в котором собрана информация о том, какие файлы будут скомпилированы, по каким путям искать заголовочные файлы и много другой информации. Впоследствии при помощи утилиты qmake из них получаются makefile для make-утилиты компилятора. Также есть возможность работы при помощи интеграторов с такими средами программирования как Microsoft Visual Studio 2003/2005 и совсем недавно стала доступна интеграция в Eclipse, для версии библиотеки 4.x.x.

Разработчики на Java могут использовать Qt с помощью фреймворка Qt Jambi от того же производителя.

При написании нового приложения часто сложно определиться, какую библиотеку удобнее использовать для построения графического интерфейса.

Настройка среды для разработки ПО на основе wxWidget

Для возможности дальнейшего бесплатного использования разрабатываемого ПО, была выбрана в качестве основы, бесплатно распространяемая среда программирования Code::Blocks 10.05.

Code::Blocks — свободная кроссплатформенная среда разработки. Code::Blocks написана на C++ и использует библиотеку wxWidgets. Имея

открытую архитектуру, может масштабироваться за счёт подключаемых модулей. Поддерживает языки программирования C, C++, D (с ограничениями).

Code::Blocks разрабатывается для Windows, Linux и Mac OS X. Среду можно собрать из исходников практически под любую Unix-подобную систему.

Для возможности разработки приложения необходимо скачать Code::Blocks без компилятора MinGW с официального сайта http://www.codeblocks.org/downloads/binaries, файл конфигурации wx-config.exe с сайта https://sites.google.com/site/wxconfig/, компилятор MinGW с сайта http://wiki.wxwidgets.org/Installing_MinGW_under_Windows, и утилиту wxPack, перейдя по ссылке https://drive.google.com/folderview?id=0B54E1bClDuSuTExoUXoyRmF0Z3c&usp=sharing.

Устанавливаем wxPack, затем MinGW, затем Code::Blocks. Утилиту wx-config.exe копируем в «C:\Windows\» или туда, где установлен windows. Процесс создания проекта с использованием библиотеки wxWidgets представлен на рисунках 1-9 соответственно.

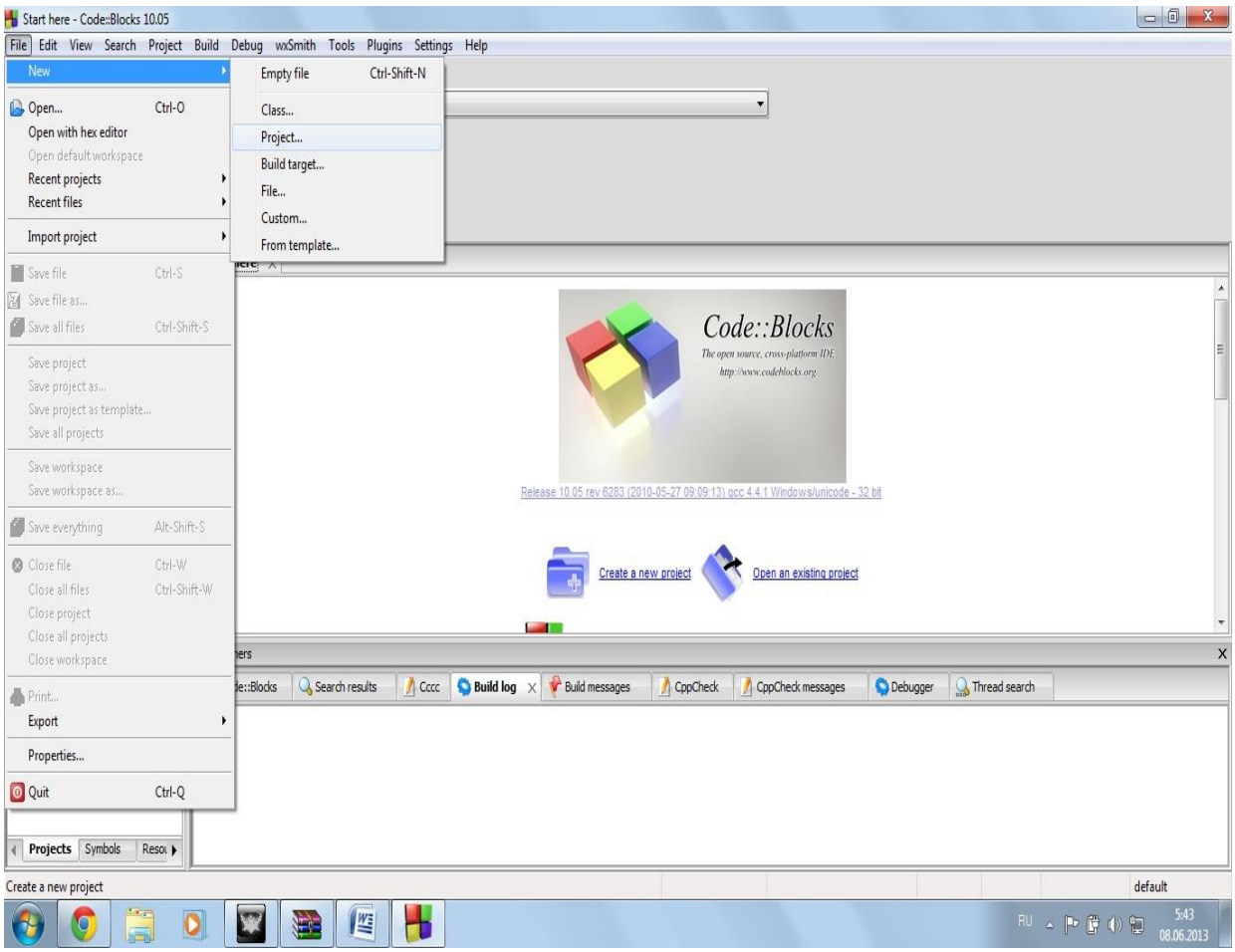
Start here - Code::Blocks 10.05
File Edit View Search Project Build Debug wxSmith Tools Plugins Settings Help
New
Open... Ctrl-O
Open with hex editor
Open default workspace
Recent projects
Recent files
Import project
Save file Ctrl-S
Save file as...
Save all files Ctrl-Shift-S
Save project
Save project as...
Save project as template...
Save all projects
Save workspace
Save workspace as...
Save everything Alt-Shift-S
Close file Ctrl-W
Close all files Ctrl-Shift-W
Close project
Close all projects
Close workspace
Print...
Export
Properties...
Quit Ctrl-Q
Empty file Ctrl-Shift-N
Class...
Project...
Build target...
File...
Custom...
From template...
Code::Blocks
The open source, cross-platform IDE
http://www.codeblocks.org
Release 10.05 rev 6283 (2010-05-27 09:09:13) gcc 4.4.1 Windows/unicode - 32 bit
Create a new project
Open an existing project
Search results
Cccc
Build log
Build messages
CppCheck
CppCheck messages
Debugger
Thread search
Projects
Symbols
Create a new project
default
RU
5:43
08.06.2013

Рис1.

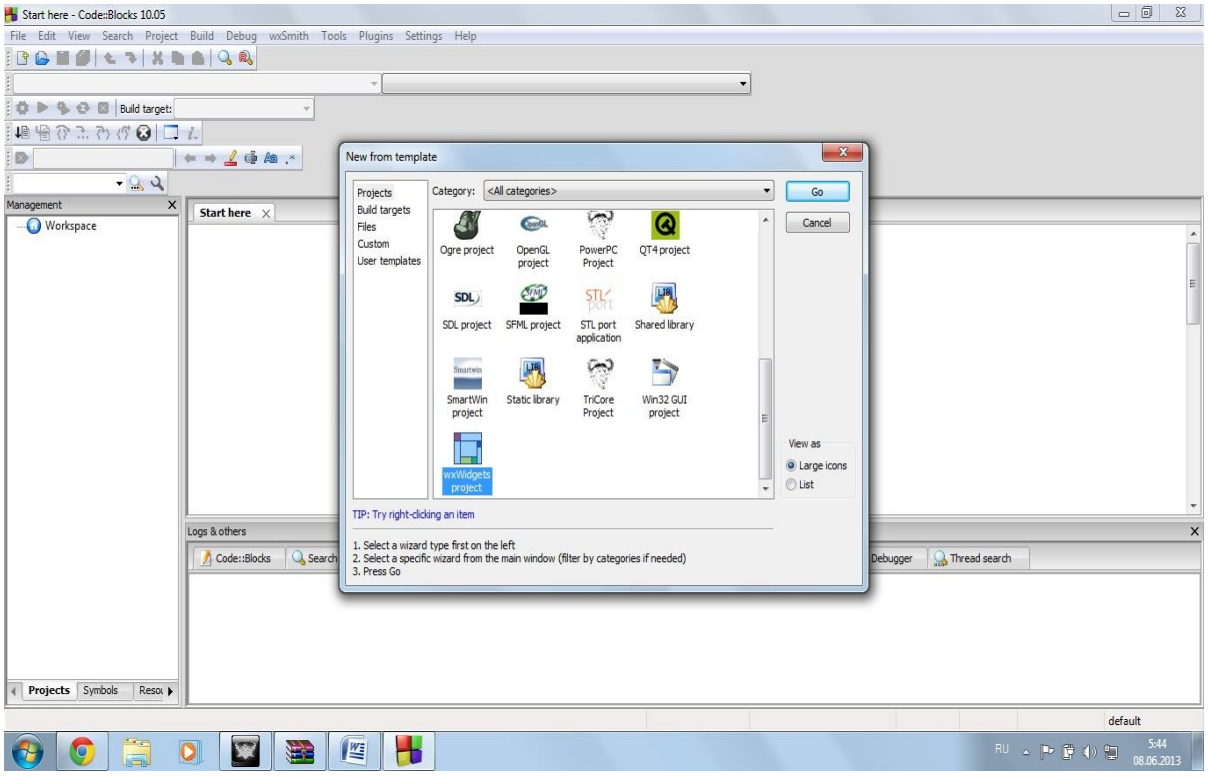
Start here - Code::Blocks 10.05
File Edit View Search Project Build Debug wxSmith Tools Plugins Settings Help
Build target:
New from template
Projects
Build targets
Files
Custom
User templates
Category: <All categories>
Go
Cancel
Ogre project
OpenGL project
PowerPC Project
QT4 project
SDL project
SFML project
STL port application
Shared library
SmartWin project
Static library
TriCore Project
Win32 GUI project
wxWidgets project
View as
Large icons
List
TIP: Try right-clicking an item
1. Select a wizard type first on the left
2. Select a specific wizard from the main window (filter by categories if needed)
3. Press Go
Management
Workspace
Start here
Logs & others
Code::Blocks
Debugger
Thread search
Projects
Symbols
default
RU
5:44
08.06.2013

Рис 2.

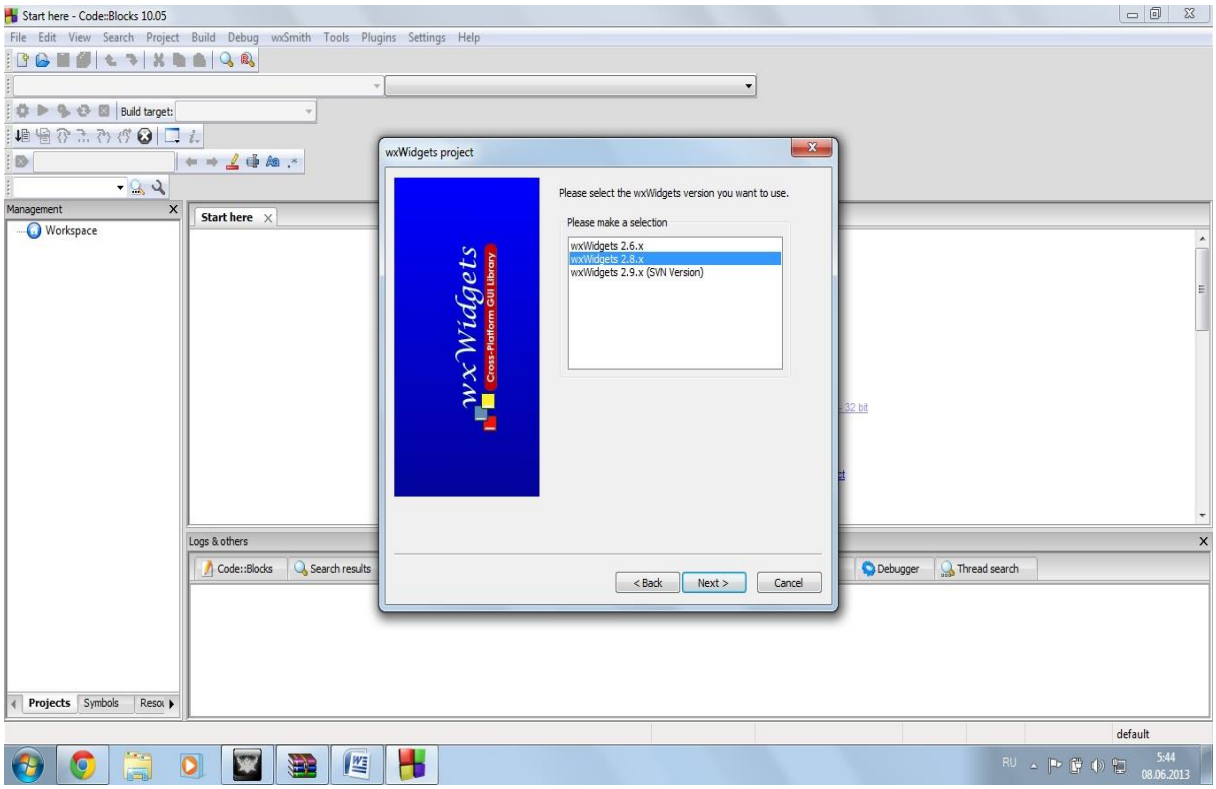
Start here - Code::Blocks 10.05
File Edit View Search Project Build Debug wxSmith Tools Plugins Settings Help
Build target:
Management
Workspace
Start here
wxWidgets project
wx Widgets
Cross-Platform GUI Library
Please select the wxWidgets version you want to use.
Please make a selection
wxWidgets 2.6.x
wxWidgets 2.8.x
wxWidgets 2.9.x (SVN Version)
< Back
Next >
Cancel
Logs & others
Code::Blocks
Search results
Debugger
Thread search
Projects
Symbols
default
RU
5:44
08.06.2013

Рис 3.

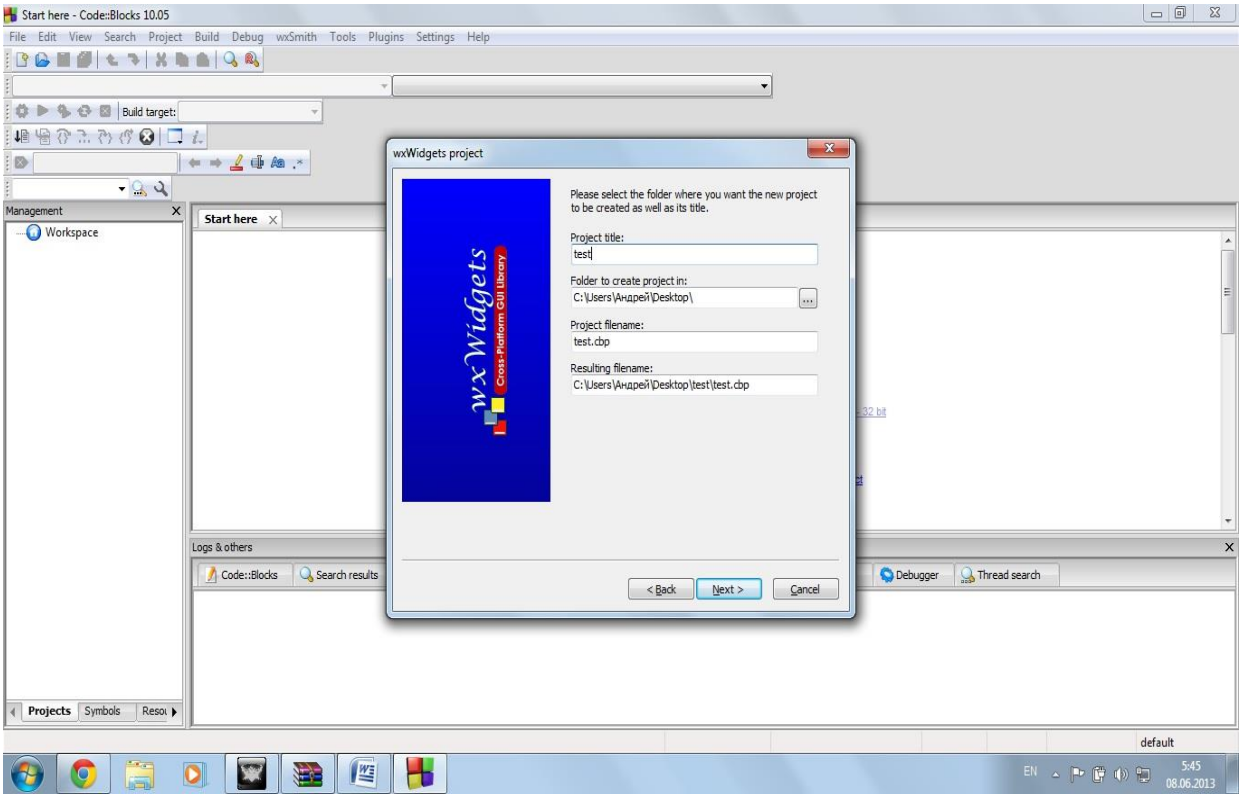
Start here - Code::Blocks 10.05
File Edit View Search Project Build Debug wxSmith Tools Plugins Settings Help
Build target:
Management
Workspace
Start here
wxWidgets project
wx Widgets
Cross-Platform GUI Library
Please select the folder where you want the new project to be created as well as its title.
Project title:
test
Folder to create project in:
C:\Users\Андрей\Desktop\
Project filename:
test.cbp
Resulting filename:
C:\Users\Андрей\Desktop\test\test.cbp
< Back
Next >
Cancel
Logs & others
Code::Blocks
Search results
Debugger
Thread search
Projects
Symbols
default
EN
5:45
08.06.2013

Рис 4.

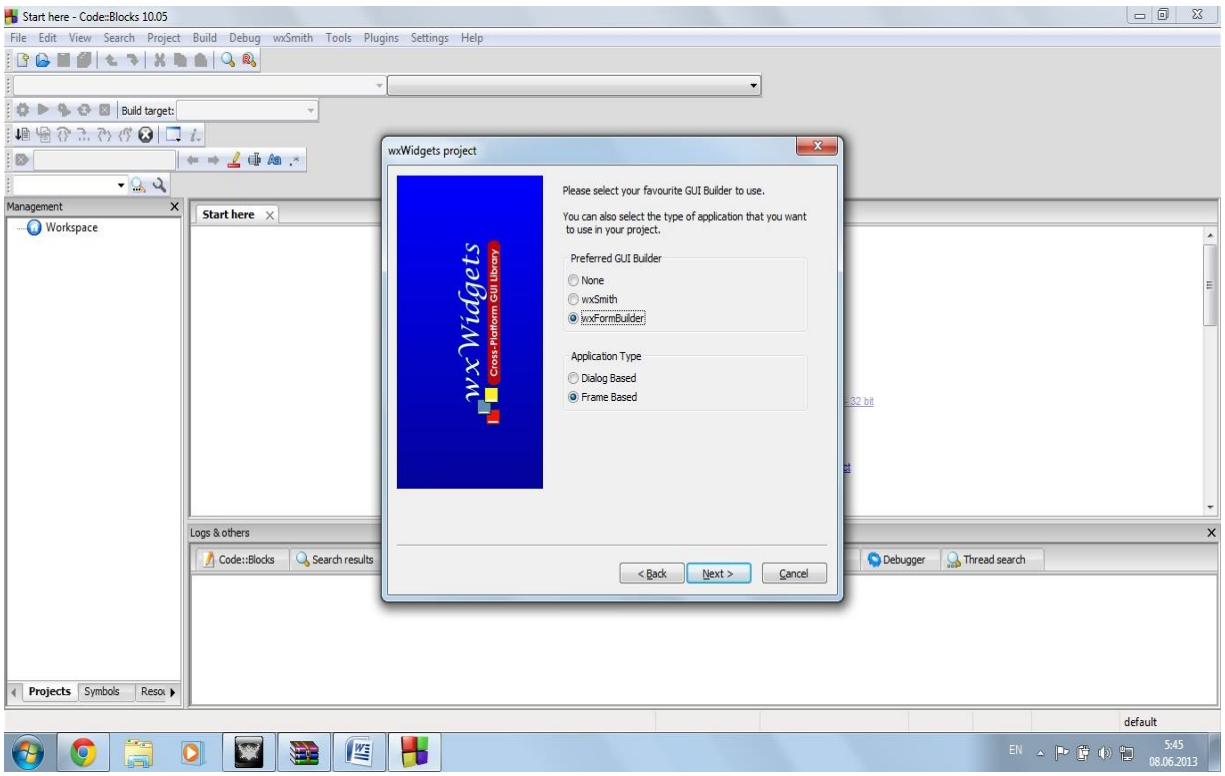
Start here - Code::Blocks 10.05
File Edit View Search Project Build Debug wxSmith Tools Plugins Settings Help
Build target:
Management
Workspace
Start here
wxWidgets project
wxWidgets
Cross-Platform GUI Library
Please select your favourite GUI Builder to use.
You can also select the type of application that you want to use in your project.
Preferred GUI Builder
None
wxSmith
wxFormBuilder
Application Type
Dialog Based
Frame Based
< Back
Next >
Cancel
Logs & others
Code::Blocks
Search results
Debugger
Thread search
Projects
Symbols
default
EN
5:45
08.06.2013

Рис 5.

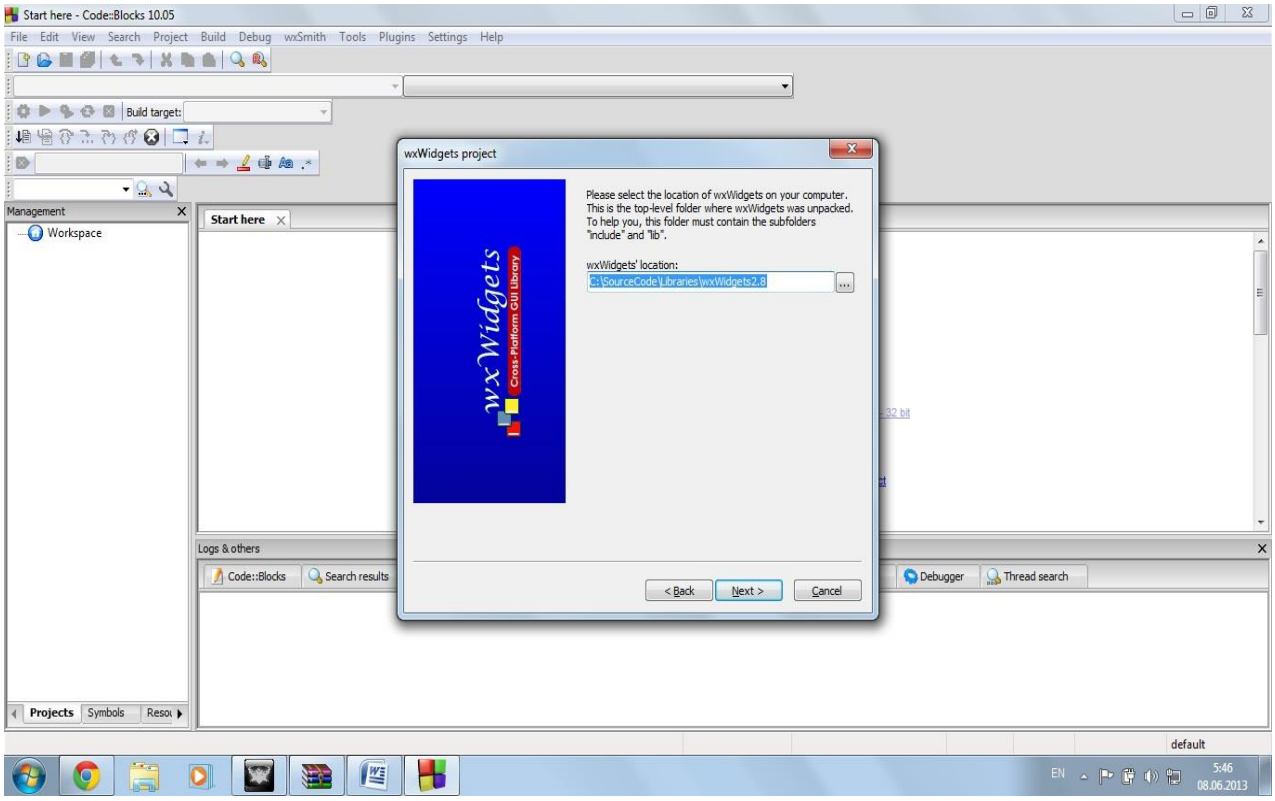
Start here - Code::Blocks 10.05
File Edit View Search Project Build Debug wxSmith Tools Plugins Settings Help
Build target:
Management
Workspace
Start here
wxWidgets project
wxWidgets
Cross-Platform GUI Library
Please select the location of wxWidgets on your computer.
This is the top-level folder where wxWidgets was unpacked.
To help you, this folder must contain the subfolders
"include" and "lib".
wxWidgets' location:
C:\SourceCode\Libraries\wxWidgets2.8
< Back
Next >
Cancel
Logs & others
Code::Blocks
Search results
Debugger
Thread search
Projects
Symbols
default
EN
5:46
08.06.2013

Рис 6.

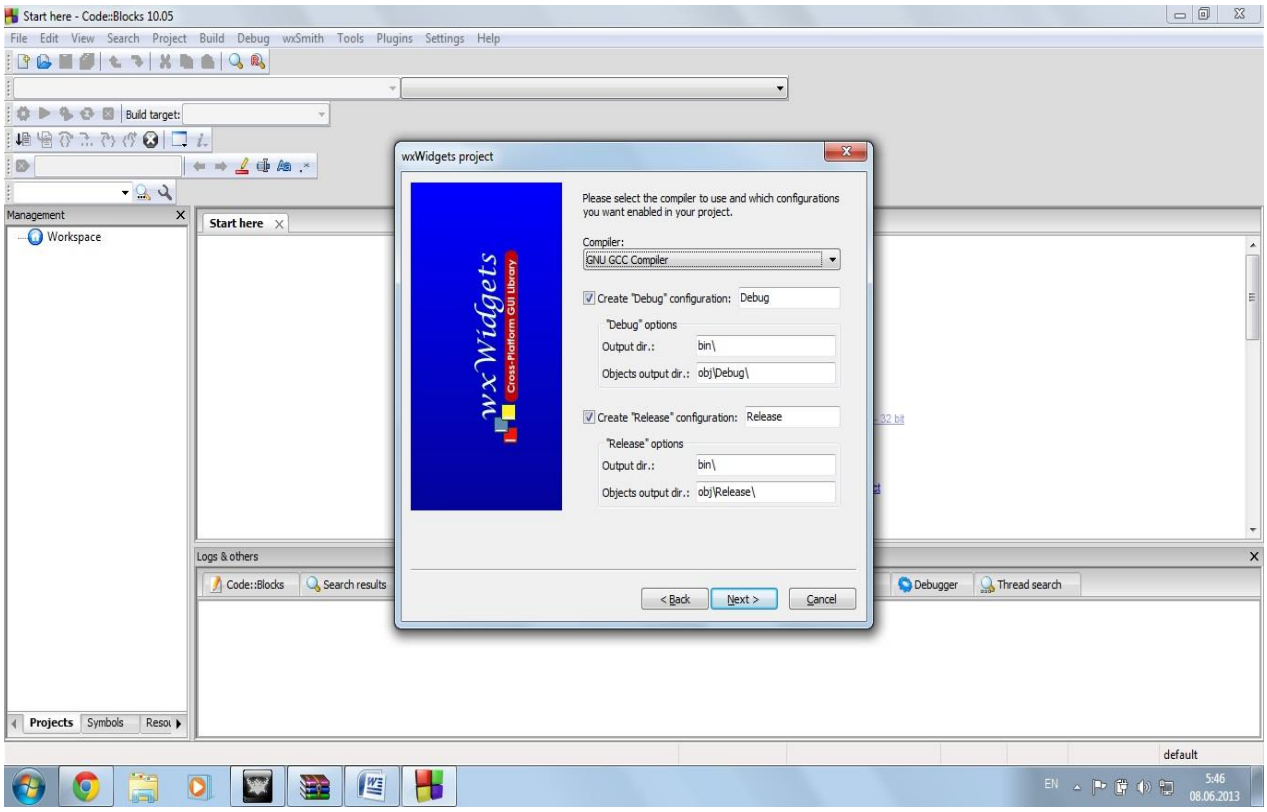
Start here - Code::Blocks 10.05
File Edit View Search Project Build Debug wxSmith Tools Plugins Settings Help
Build target:
Management
Workspace
Start here
wxWidgets project
wx Widgets
Cross-Platform GUI Library
Please select the compiler to use and which configurations you want enabled in your project.
Compiler:
GNU GCC Compiler
Create "Debug" configuration:
Debug
"Debug" options
Output dir.:
bin\
Objects output dir.:
obj\Debug\
Create "Release" configuration:
Release
"Release" options
Output dir.:
bin\
Objects output dir.:
obj\Release\
< Back
Next >
Cancel
Logs & others
Code::Blocks
Search results
Debugger
Thread search
Projects
Symbols
default
EN
5:46
08.06.2013

Рис 7.

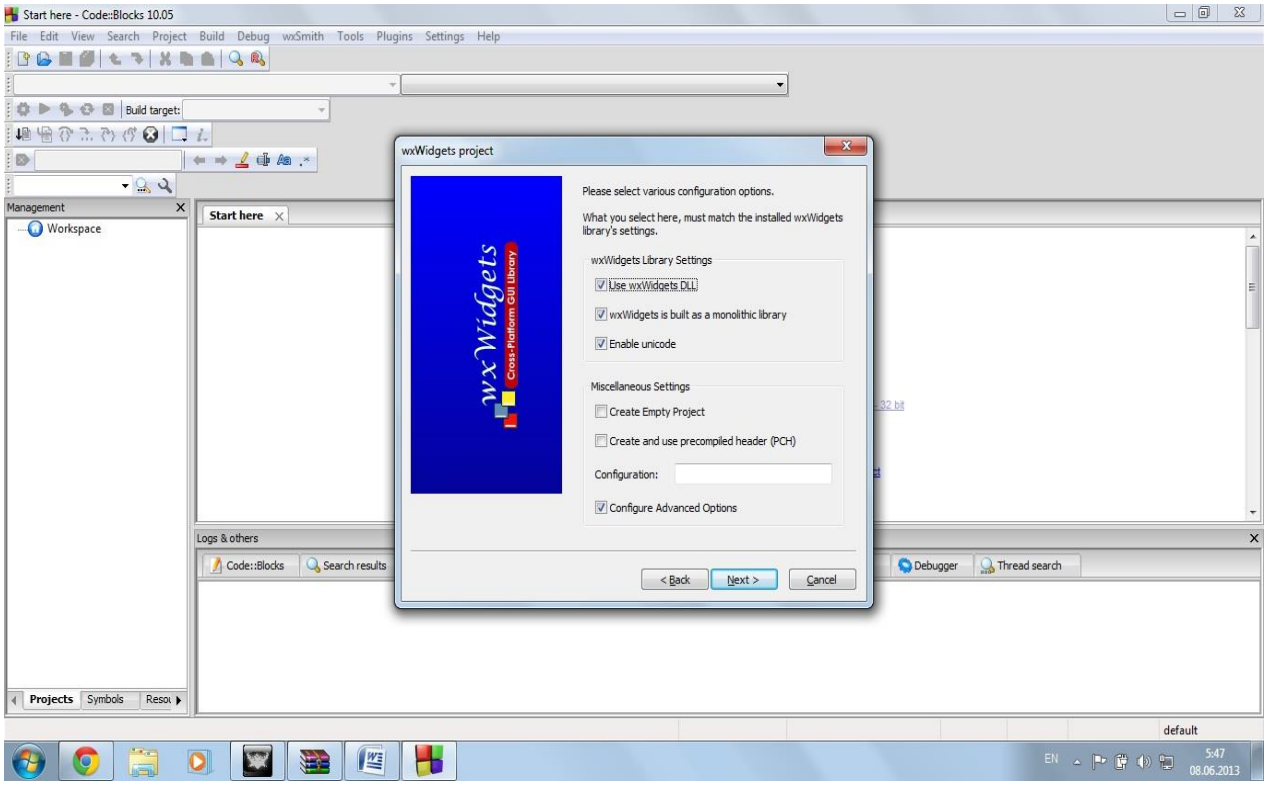
Start here - Code::Blocks 10.05
File Edit View Search Project Build Debug wxSmith Tools Plugins Settings Help
Build target:
Management
Workspace
Start here
wxWidgets project
wx Widgets
Cross-Platform GUI Library
Please select various configuration options.
What you select here, must match the installed wxWidgets library's settings.
wxWidgets Library Settings
Use wxWidgets DLL
wxWidgets is built as a monolithic library
Enable unicode
Miscellaneous Settings
Create Empty Project
Create and use precompiled header (PCH)
Configuration:
Configure Advanced Options
< Back
Next >
Cancel
Logs & others
Code::Blocks
Search results
Debugger
Thread search
Projects
Symbols
default
EN
5:47
08.06.2013

Рис 8.

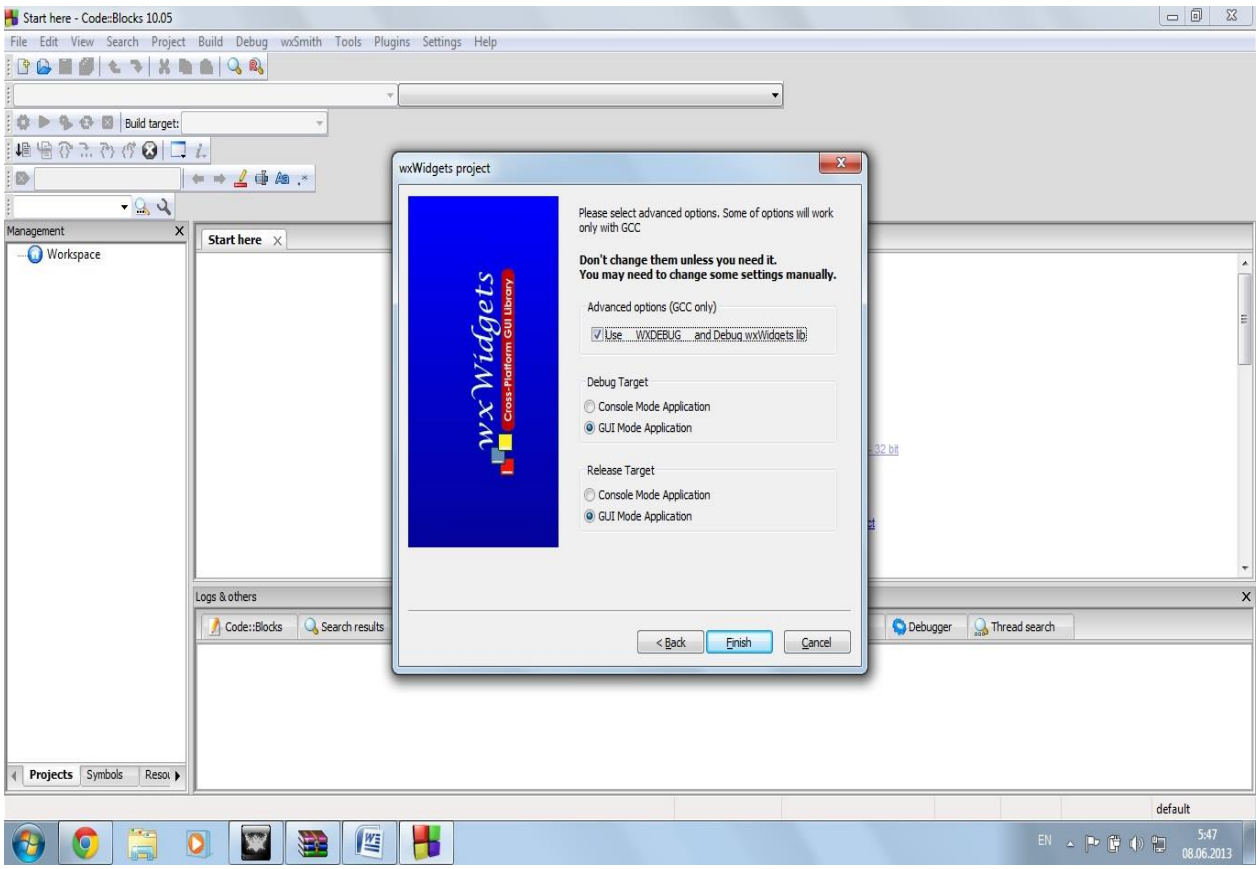

Рис 9.

Лабораторные работы.

Лабораторная работа №1

Диалоги.

Задание.

Необходимо создать простейшее приложение с использованием библиотеки wxWidgets, которое показывает работу с диалоговыми окнами согласно варианту задания.

Справочный материал

Диалоговые окна или диалоги необходимая часть большинства современных GUI приложений. Диалог определяет общение между двумя и более персонами. В приложении диалог - это окно, которое используется для «общения» с программой. Диалог используется для ввода данных, их

изменения, изменения настроек приложения и т.д. Диалоги - это важное средство для коммуникации между пользователем и программой.

Существует два типа диалогов: предопределённые и пользовательские.

Предопределённые диалоги.

Предопределённые диалоги - это диалоги доступные в наборе инструментов wxWidgets (wxWidgets toolkit). Здесь есть диалоги для различных целей, например, вывод текста, получение данных, открытие и сохранение файлов и т.д. Всё это помогает программисту сэкономить время и повысить производительность труда.

Каждое приложение на wxWidgets определяет свой собственный класс приложения, являющийся потомком от wxApp. В программе существует единственный экземпляр данного класса, который представляет из себя представление данного приложения. Обычно этот класс объявляет функцию OnInit, которая вызывается, когда библиотека wxWidgets готова запустить ваш код (функция является эквивалентом функции main языка C или WinMain в Win32-приложениях). Ниже дано минимально возможное объявление класса приложения:

```
// Объявляем класс приложения
class MyApp : public wxApp
{
public:
// Вызывается при старте приложения
virtual bool OnInit();
};
```

Реализация функции OnInit обычно создает по крайней мере одно окно, считывает параметры из командной строки, инициализирует нужные для работы структуры данных и выполняет другие действия, необходимые для запуска программы. Если функция возвращает true, то wxWidgets запускает петлю событий, которая обрабатывает ввод пользователя и запускает соответствующий обработчик событий в случае необходимости. Если данная

функция возвращает false, то wxWidgets корректно очищает свои внутренние структуры и завершает работу приложения. Простейшая реализация функции OnInit может иметь следующий вид:

```
// Инициализируем приложение
bool MyApp::OnInit()
{
// Создаем главное окно приложения
MyFrame *frame = new MyFrame(wxT("Minimal wxWidgets App"));
// Показываем его
frame->Show(true);
// Запускаем петлю событий
return true;
}
```

Реализация создает экземпляр нового класса окна MyFrame (мы определим этот класс позднее), показывает окно на экране и возвращает true, чтобы запустить петлю событий. Главные окна (такие как фреймы и диалоги), в отличие от дочерних, должны быть показаны сразу после создания. Заголовок главного окна передается конструктору, обернутым в макрос wxT(). В дальнейшем вы заметите, что этот макрос используется в большинстве примеров библиотеки wxWidgets и позволяет преобразовать строки и символы к правильномутипу, что позволяет приложению компилироваться в режиме Unicode. Данный макрос является синонимом макроса _T(). Также в программах вы возможно встретите макрос _(), который говорит библиотеке, что строку можно перевести на другой язык.

Где же находится код, который создает экземпляр класса MyApp? wxWidgets делает это самостоятельно, но вы должны явно указать тип объекта, который необходимо создать. Поэтому вам необходимо добавить следующие строки в ваш файл с реализацией:

```
// Говорит wxWidgets, что надо создать объект MyApp
IMPLEMENT_APP(MyApp)
```

Без этого определения wxWidgets не сможет создать объект вашего приложения. Данный макрос также вставляет код, который проверяет, что объект является приложением и библиотека была скомпилирована с необходимыми опциями. Это позволяет wxWidgets сообщать о некоторых ошибках, которые впоследствии могли бы вызвать появление трудно выявляемых ошибок времени выполнения. Когда wxWidgets создает объект типа MyApp, результат сохраняется в глобальной переменной wxTheApp. Можно использовать эту переменную в вашей программе, однако лучше явно не обращаться к ней в своем коде. Вместо этого, после объявления вашего класса приложения поместите макрос

```
// Реализация MyApp& GetApp()
DECLARE_APP(MyApp)
```

после чего вы сможете использовать в программе функцию wxGetApp, которая возвращает ссылку на объект приложения типа MyApp.

Простая кнопка

В следующем примере мы создадим кнопку в рамке виджета. А также простой обработчик событий.

```
#include <wx/wx.h>
class Button : public wxFrame
{
        public:
                Button(const wxString& title);

                void OnQuit(wxCommandEvent & event);
};

#include "button.h"

Button::Button(const wxString& title)
        : wxFrame(NULL, wxID_ANY, title, wxDefaultPosition, wxSize(270, 150))
```

```
{
	wxPanel *panel = new wxPanel(this, wxID_ANY);

	wxButton *button = new wxButton(panel, wxID_EXIT, wxT("Quit"),
		wxPoint(20, 20));
	Connect(wxID_EXIT, wxEVT_COMMAND_BUTTON_CLICKED,
		wxCommandEventHandler(Button::OnQuit));

	button->SetFocus();

	Centre();
}

void Button::OnQuit(wxCommandEvent & WXUNUSED(event))
{
	Close(true);
};

#include <wx/wx.h>

class MyApp : public wxApp
{
	public:
		virtual bool OnInit();
};

#include "main.h"
#include "button.h"

IMPLEMENT_APP(MyApp)
```

```
bool MyApp::OnInit()
{
	Button *btnapp = new Button(wxT("Button"));
	btnapp->Show(true);

	return true;
};
```

Внутри метода OnQuit() мы вызываем метод Close(), который прекращает работу программы.

Варианты заданий

Вариант 1

Создать приложение с 2 кнопками, которые выводят на 2 различных диалога.

Вариант 2

Создать приложение с кнопкой, которая организует выход из приложения.

Вариант 3

Создать приложение, в процессе работы которого выводится диалоговое окно, сообщающее об ошибке, в информирующем окне должна быть кнопка, корректно завершающая работу приложения.

Лабораторная работа № 2

Окна.

Задание.

Изучить свойства и приемы работы с окнами в wxWidgets.

Справочный материал

Концепция окна

Окно это прямоугольная область с общим набором свойств: его размеры можно изменять, оно может нарисовать себя, его можно показать или

спрятать и т.д.. Оно может содержать в себе другие окна (такие как фрейм с полосой меню — menu bar, панель инструментов – toolbar, панель состояния — status bar) или не дочерние окна (такие как статичный текст — static text control). Обычно, окно, которое вы видите на экране в приложении wxWidgets владеет соответствующим объектом класса wxWindow или производным от него, но это не всегда так. Например, раскрывающийся список (drop down list) в местном wxComboBox обычно не моделируется отдельным wxWindow.

Клиентская и не клиентская область

Когда мы ссылаемся на размер окна, мы обычно включаем туда и внешние измерения, декорации, такие как рамка и заголовок. Когда же мы ссылаемся на размер клиентской области окна, мы подразумеваем область внутри окна на которую или в которую могут быть помещены дочерние окна. Клиентская область не включает пространство которое занято полосой меню, панелью состояния и панелью инструментов.

Полосы прокрутки

Большинство окон способны изображать полосы прокрутки, управляемые этим же окном , а не добавленные приложением явно. Клиентская область, в таком слуаче, уменьшается на пространство используемое этой полосой прокрутки. Для оптимизации, только окнам со стилями wxHSCROLL и wxVSCROLL гарантируется владение их собственными полосами проктуки. Больше информации по скроллингу можно найти в этой главе дальше, где мы обсуждаем wxScrollWindow.

Каретка и курсор

Окно может иметь один экземпляр wxCaret (для отображения текущей позиции текста) и один экземпляр курсора (для отображения текущей позиции мышки). Когда мышка переходит в окно, wxWidgets автоматически отображает тот курсор, который был установлен для этого окна. Когда окошко получает фокусировку (focus), то каретка (если есть) будет показана

на её текущей позиции, или на том месте, куда вы щелкнули кнопкой мышки при передаче фокусировки этому окну.

Верхнеуровневые окна

Окна резко делятся на верхнеуровневые окна (wxFrame, wxDialog, wxPopup) и на другие окна. Только верхнеуровневые окна могут быть созданы от NULL родителя и только верхнеуровневые окна имеют отложенное разрушение (они не удаляются до тех пор пока все остальные события (events) не будут обработаны). Исключая всплывающие (pop-up) окна, верхнеуровневые окна могут иметь декорации вроде заголовков (title bars) и закрывающих кнопок. Обычно их можно таскать по экрану и менять их размеры, если приложение позволяет.

Система координат

Система координат всегда начинается с (0,0) от верхнего-левого угла, окно измеряется в пикселях. Источник и масштаб могут варьироваться в зависимости от контекста устройства (device context) который используется для отрисовки окна.

Рисование

Когда окну требуется быть нарисованным оно получает два события: wxEVT_ERASE_BACKGROUND для отрисовки фона и wxEVT_PAINT для для отрисовки переднего плана. Готовый к использованию классы, типа wxButton, рисуют сами себя, но для того чтобы создать свой особый класс окна Вам понадобиться управлять этими событиями. Вы можете оптимизировать отрисовку, получая область обновления (ту часть, которую надо обновить) от окна.

Цвет и шрифт

Каждое окно имеет настройки цвета фона и переднего плана которые могут быть использованы чтобы определить цвет фона и (обычно реже) цвет переднего плана. Обработчик очистки фона по умолчанию использует цвет фона окна или, если он не был установлен, то соответствующий цвет или

текстуру текущей цветовой схемы или темы. Окно также имеет настройки шрифта, которые могут использоваться или нет в зависимости от типа окна.

Окно Variant

В MAC OS X окно имеет концепцию «вариант окна» (window variant), таким образом оно может быть показано в зависимости от размеров: wxWINDOW_VARIANT_NORMAL (по умолчанию), wxWINDOW_ VARIANT_SMALL, wxWINDOW_VARIANT_MINI, or wxWINDOW_VARIANT_LARGE. Изменение к меньшему варианту полезны, когда у вас много информации для отображения а пространство ограниченно, но этим следует пользоваться умеренно. Некоторые приложения используют малый вариант повсеместно.

Изменение размеров

Когда изменяются размеры окна, приложением или пользователем, оно получает событие wxEVT_SIZE. Если у окна есть дети, они так же должны быть соответственно позиционированы и изменены в размерах, рекомендуется использовать калибровщики (sizers), они обсуждаются в главе 7, «Расположение окна используя калибровщики». Большая часть из набора окон имеет представление о размере и позиции по умолчанию если Вы передадите им wxDefaultSize или wxDefaultPosition (или -1 как персональный размер или значение позиции). С этой целью, каждый элемент управления реализует DoGetBestSize, который возвращает разумный размер по умолчанию основанный на содержимом элемента управления, текущем шрифте и других факторах.

Ввод

Любое окно может получить входящий сигнал от мышки в любое время, если только, какое нибудь другое окно не захватило мышку на время или если окно неисправно или намеренно отключено. Только окошко с фокусировкой может получать ввод с клавиатуры. Приложение может установить фокусировку окна самостоятельно, wxWidgets так же устанавливает фокусировку на окно когда пользователь щелкает на нём кнопкой мыши.

Окну, которое получает фокусировку, посылается сообщение wxEVT_SET_FOCUS. Сообщение wxEVT_KILL_FOCUS посылается когда фокусировка передается другому окну.

Обработка времени простоя и обновление интерфейса пользователя

Все окна являются (если только не указанно иное) получателями событий простоя (idle events: wxEVT_IDLE) которые посылаются когда все другие события были обработаны, заданные при помощи EVT_IDLE(func) макрос.

Обновление интерфейса пользователя это особый вид обработки времени простоя, в которому все окна могут задать функцию которая обновляет состояние окна. Эта функция вызывается периодически во время простоя. В дальнейшем описании событий EVT_UPDATE_UI(id, func) обычно опускается для краткости.

Создание и удаление окна

Большинство оконных классов позволяют создание в один или два шага. wxButton имеет типичную сигнатуру конструктора.

```
wxButton();
wxButton(wxWindow* parent,
wxWindowID id,
const wxString& label = wxEmptyString,
const wxPoint& pos = wxDefaultPosition,
const wxSize& size = wxDefaultSize,
long style = 0,
const wxValidator& validator = wxDefaultValidator,
const wxString& name = wxT(“button”));
```

Следующий пример создания в один шаг использует значения по умолчанию из конструктора.

```
wxButton* button = new wxButton(parent, wxID_OK);
```

Вы обязаны передать не NULL родительское окно в конструктор если только это не фрейм(окно с рамкой) или не диалог. Это автоматически добавит дочернее окно в родителя и, когда родитель разрушается, дети будут также

разрушены. Как мы видели ранее, Вы передаёте стандартный или определённый пользователем идентификатор в окно для того чтобы однозначно идентифицировать его. Вы можете также передать wxID_ANY, и wxWidgets сгенерирует подходящий идентификатор (отрицательное значение, чтобы отличить его от определённых пользователем или стандартных идентификаторов). Вы можете передать позицию и размер окна, валидатор, если есть подходящий (см. главу 9), стиль (см. следующий абзац) и строку с именем. Вы можете передать произвольную строку в качестве имени или проигнорировать её, это используется редко но было добавлено чтобы позволить подгонку окон в Xt и Motiff, которые требуют идентификации элемента управления по имени.

Для создания в два шага Вы используете конструктор по умолчанию и затем вызываете Create, которая имеет такую же сигнатуру как и полноценный конструктор. Пример создания в два шага:

```
wxButton* button = new wxButton;
button->Create(parent, wxID_OK);
```

Только когда вы вызываете Create создается подразумеваемое окно и событие wxEVT_CREATE посылается в окно если, требуется дальнейшая обработка, на этой стадии.

Зачем бы вам использовать создание в два шага? Одна причина в том что вы можете захотеть отложить полное создание окна до тех пор, пока оно действительно не понадобиться. Другая в том чтобы установить некие свойства окна перед вызовом Create, особенно это касается тех свойств, которые используются внутри вызова Create. Например, вы хотите установить дополнительный стиль wxWS_EX_VALIDATE_RECURSIVELY (который должен быть установлен посредством SetExtraStyle). При создании диалога, этот стиль используется внутри функции Create для первичного контроль (initial validation), таким образом, если Вам это требуется, то важно установить этот дополнительный стиль до создания диалога.

Когда вы создаёте wxWindow, или не верхне-уровневый унаследованный оконный класс, он всегда в состоянии «виден», так что если родитель виден в момент создания то ваше окно тоже будет видно. Затем, вы сможете использовать Show(false) чтобы спрятать окно, если потребуется. Это отличие от wxDialog и wxFrame, которые изначально создаются в состоянии «спрятанно» чтобы дать возможно приложению расположить элементы управления без мельканий, перед тем как показывать посредством Show или ShowModal (для модальных диалогов).

Окна уничтожаются посредством вызова Destroy (для верхне-уровневых окон) или delete (для дочерних окон), сообщение wxEVT_DESTROY посылается прямо перед тем как разрушается реальное окно. На самом деле, дочерние удаляются автоматически, таким образом он редко удаляются явно.

Стили окна

Окно имеет стиль и дополнительный стиль. Стили окна это лаконичный путь указать альтернативное поведение и внешний вид для окон в момент создания. Символы определены таким образом что-бы их можно было комбинировать в набор битов, используя побитовый оператор C++ «или». Например:

wxCAPTION | wxMINIMIZE_BOX | wxMAXIMIZE_BOX | wxTHICK_FRAME

Класс wxWindow имеет базовый набор стилей, таких как стиль с рамкой. Каждый последующий, в иерархии наследования, класс может добавлять стили. Дополнительный стиль (“extra” style) размещает значения которые нельзя подогнать в значение стиля.

Базовые классы окна

Эти классы реализуют функциональность для конкретных классов потомков.

wxWindow — базовый класс для всех окон.

wxControl — базовый класс для всех элементов управления, таких, как wxButton.

wxControlWithItems — базовый класс для многоэлементных элементов управления..

Верхнеуровневые окна

Верхнеуровневые окна обычно существуют независимо на рабочей области.

wxFrame — окно с изменяемыми размерами, которое содержит в себе другие окна. wxMDIParentFrame — фрейм, который управляет другими фреймами.

wxMDIChildFrame — фрейм, которым управляет фрейм родитель.

wxDialog — окно, с изменяемыми размерами, для представления вариантов выбора.

wxPopupWindow — временное окно с минимальным декорированием.

Окна – контейнеры

Окна – контейнеры управляют дочерними окнами.

wxPanel — окно для расположения элементов управления.

wxNoteBook — окно, для переключения страниц при помощи вкладок.

wxScrolledWindow — окно, которое прокручивает потомков и графику.

wxSplitterWindow — окно, которое управляет двумя дочерними окнами.

Не статичные элементы управления

Пользователь может редактировать эти контроллеры.

wxButton — кнопка с текстовой меткой.

wxBitmapButton — кнопка с растровым изображением.

wxChoice — раскрывающийся список с вариантами выбора.

wxComboBox — поле со списком с вариантами выбора и с возможностью редактирования

wxCheckBox — окно, представляющее кнопку-флажок, вкл. или выкл.

wxListBox — список строковых элементов с возможность выбора.

wxRadioBox — сетка из переключателей.

wxRadioButton — элемент управления напоминающий переключатель.

wxScrollBar — полоса прокрутки

wxSpinButton — стрелочки для увеличения/уменьшения значений.

wxSpinCtrl — текстовое поле и стрелочки для редактирования целых значения.

wxSlider — контроллер, для изменения значения в данном диапазоне.

wxTextCtrl — одно- или многострочное поле для ввода текста

wxToggleButton — кнопка тумблер, которая может быть включена или выключена.

Статичные контроллеры

Эти элементы управления представляют информацию и не могут быть изменены пользователем.

wxGauge — контроллер, показывающий количество или прогресс.

wxStaticText — контроллер, показывающий текстовую метку.

wxStaticBitmap — контроллер, который изображает растровое изображение.

wxStaticLine — контроллер, отображающий линию.

Меню

Меню, это временные окна, содержащие в себе списки команд.

wxMenu — может быть использовано как контекстное меню или в панели меню.

Панели управления

Панели управления представляют быстрый доступ к командам и информации, обычно внутри wxFrame.

wxMenuBar — панель меню, которая представляет команды в wxFrame.

wxToolBar — панель инструментов, которая представляет быстрый доступ к командам.

wxStatusBar — панель состояния, которая отображает информацию в нескольких полях.

Базовые оконные классы

wxWindow это и важный базовый класс и конкретный класс окна, который вы можете инстанцировать. Однако, скорее всего, вы будете наследовать от него (или использовать уже существующих наследников) а не пользоваться

им напрямую. Как мы уже видели, wxWindow может быть создан или в один шаг, используя конструктор не по умолчанию, или в два шага, используя конструктор по умолчанию и следом Create. Для создания в один шаг, пользуйтесь этим конструктором.

```
wxWindow(wxWindow* parent,
   wxWindowID id,
   const wxPoint& pos = wxDefaultPosition,
   const wxSize& size = wxDefaultSize,
   long style = 0,
   const wxString& name = wxT("panel"));
```

Например:

```
wxWindow* win  = new wxWindow(parent, wxID_ANY,
   wxPoint(100, 100), wxSize(200, 200));
```

Стили wxWindow

Каждый оконный класс может дополнять базовые стили, определённые для wxWidow и перечисленные в таблице 4-1. Не все местные контроллеры поддерживают все стили рамок, и если рамка не указанна, то будет использован соответствующий стиль по умолчанию. Например, в ОС Windows, большинство wxControl, это унаследованные классы, использующие wxSUNKEN_BORDER по умолчанию, который будет интерпретироваться как стиль рамки для текущей темы. Приложение может подавить рамку по умолчанию используя такой стиль как wxNO_BORDER.

Базовые стили окна

wxSIMPLE_BORDER Отображает тоненький бордюрчик вокруг окна

wxDOUBLE_BORDER Отображает двойной бордюр

wxSUNKEN_BORDER Отображает утопленный бордюр или бордюр контроллера согласующийся с текущей темой.

wxRAISED_BORDER Отображает выпуклый бордюр.

wxSTATIC_BORDER Отображает бордюр, подходящий для статичного контрола. Только для ОС Windows.

wxNO_BORDER Отображает без бордюра. Таким образом отменяет любые попытки wxWidgets добавить подходящий бордюр.

wxTRANSPARENT_WINDOW Определяет прозрачное окно (то которое не получает события отрисовки. Только для ОС Windows.

wxTAB_TRAVERSAL Используйте это чтобы включить переходы по Tab для не диалоговых окон.

wxWANTS_CHARS Пользуйтесь этим чтобы обозначить что это окно желает получать все события от клавиатуры, даже от клавиш Tab и Enter, которые используются для диалоговой навигации и которые не сгенерировались без этого стиля.

wxFULL_REPAINT_ON_RESIZE По умолчанию, под ОС Winows, wxWidgets не будет перерисовывать клиентскую область целиком во время изменения размера. Этот стиль гарантирует что вся клиентская область будет перерисована вся клиентская область.

wxVSCROLL Включает вертикальную полосу прокрутки.

wxHSCROLL Включает горизонтальную полосу прокрутки.

wxALWAYS_SHOW_SB Если у окна есть полосы прокрутки, то оно просто отключает их, когда в них нет нужды, вместо того, чтобы прятать (когда размер окна достаточно велик и для навигации по нему полоса прокрутки не требуется). Этот стиль, в момент написания книги, реализован только для ОС Windows и wxUniversal.

wxCLIP_CHILDREN Только для ОС Windows. Используется для того, чтобы устранить мерцание, вызванное тем, что окно очищает фон своих дочерних окон.

Основные дополнительные стили окна.

wxWS_EX_VALIDATE_RECURSIVELY По умолчанию, Validate, transferDataToWindow, и transferDataFromWindow работает только на прямых потомков окна. Устанавливайте этот стиль если хотите сделать их рекурсивно спускающимися по всем под-окнам.

wxWS_EX_BLOCK_EVENTS wxCommandEvents и объект наследника отправляются к родительскому окну, и так далее, рекурсивно, по умолчанию Использую, этот стиль, для данного окна, включает возможность блокировать это распространение в этом окне, ради того, чтобы предотвратить распространение событий далее. У диалогов такой стиль по умолчанию, но заметьте, что если SetExtraStyle вызывается приложением, то он может быть переустановлен.

wxWS_EX_TRANSIENT Не используйте такое окно как неявного родителя для других окон. Этот стиль должен быть использован для временных окон, иначе, есть риск создать диалог или фрейм с таким окном в качестве родителя, что может привести к краху, если родитель будет уничтожен прежде потомка.

wxWS_EX_PROCESS_IDLE Этому окну следует всегда обрабатывать события простоя, даже если режим, установлен wxIdleEvent::SetMode как wxIDLE_PROCESS_SPECIFIED.

wxWS_EX_PROCESS_UI_UPDATES Этому окну следует всегда обрабатывать обновления событий пользовательского интерфейса, даже если режим, установлен wxUpdateUIEvent::SetMode как wxUPDATE_UI_PROCESS_SPECIFIED. См. Главу 9 за дальнейшей информацией по событиям обновления пользовательского интерфейса.

События wxWindow

EVT_WINDOW_CREATE(func) Обрабатывает распространение события wxEVT_CREATE, генерируемое когда низлежащие окна были только что созданы. Обработчики получают объект wxWindowCreateEvent.

EVT_WINDOW_DESTROY(func) Обрабатывает распространение события wxEVT_DELETE, генерируемое когда окно вот-вот будет разрушено. Обработчики получают объект wxWindowDestroyEvent.

EVT_PAINT(func) Обрабатывает событие wxEVT_PAINT, генерируемое когда окну требуется обновление. Обработчики получают объект wxPaintEvent.

EVT_ERASE_BACKGROUND(func) Обрабатывает событие wxEVT_ERASE_BACKGROUND, генерируемое когда окну требуется обновить фон. Обработчики получают объект wxEraseEvent.

EVT_MOVE(func) Обрабатывает событие wxEVT_MOVE, генерируемое когда окно движется. Обработчики получают wxMoveEvent объект .

EVT_SIZE(func) Обрабатывает событие wxEVT_SIZE, генерируемое когда окно меняет свои размеры. Обработчики получают объект wxSizeEvent.

EVT_SET_FOCUS(func) \ EVT_KILL_FOCUS(func) Обрабатывает события wxEVT_SET_FOCUS и wxEVT_KILL_ FOCUS, генерируемые когда окну поучает или теряет фокусировку клавиатуры. Обработчики получают объект wxFocusEvent.

EVT_SYS_COLOUR_CHANGED(func) Обрабатывает событие wxEVT_SYS_COLOUR_CHANGED, когда пользователь сменил цвет в панели управления (только для ОС Windows). Обработчики получают объект wxSysColourChangedEvent.

EVT_IDLE(func) Обрабатывает событие wxEVT_IDLE, генерируемое во время простоя. Обработчики получают wxIdleEvent объект .

EVT_UPDATE_UI(func) Обрабатывает событие wxEVT_UPDATE_UI, генерируемое во время простоя, чтобы дать окну возможность обновить себя.

Варианты заданий

Вариант 1

Создать окно в котором должно быть меню с кнопками exit(осуществляет выход) и about(выводит сообщение о программе).

Вариант 2

Создать окно с кнопкой, при нажатии на которую, размер окна изменяется.

Вариант 3

Создать главное окно с кнопкой, при нажатии на которую, появляется дочернее окно.

Лабораторная работа №3

Рисование в wxWidgets.

Задание

Изучить графические возможности и классы для рисования в wxWidgets.

Справочный материал

Код для рисования в wxWidgets работает как очень быстрый художник: мгновенно выбирает цвет и инструмент, рисует маленькую часть картины, далее выбирает другой инструмент и рисует другую часть картины и так далее. В данном разделе описываются инструменты для рисования: классы wxColour, wxPen, wxBrush, wxFont и wxPalette. Класс wxColour необходимо использовать, чтобы определить различные параметры цвета во время рисования (из-за того, что проект wxWidgets родился в Эдинбурге его API использует британское написание слова «цвет», однако позже для соответствия более распространенному американскому языку в wxWidgets был определен wxColor как синоним для wxColour). Вы можете установить цвет фона и символов текста для контекста устройства, используя методы SetTextForeground и SetTextBackground. Также можно передать параметр класса wxColour при создании кистей и перьев. Объект класса wxColour можно создать множеством различных способов. Вы можете передать интенсивность красного, зеленого и синего (значение от 0 до 255), или стандартную строку с цветом (такую как WHITE или CYAN). Также можно создать цвет из другого объекта wxColour. Кроме того, можно использовать предопределенные объекты, которые являются указателями на цветовые объекты:

wxBLACK, wxWHITE, wxRED, wxBLUE, wxGREEN, wxCYAN и wxLIGHT_GREY. Существует объект wxNullColour, который по сути является неинициализированным цветом для которого метод Ok всегда возвращает false.

Используя класс wxSystemSettings можно получить некоторые стандартные системные цвета, такие цвет обычной 3D-поверхности, обычный цвет фона

окна, цвет текста меню и так далее. Вот различные примеры создания объектов wxColour:

```
wxColour color(0, 255, 0); // зеленый
wxColour color(wxT("RED")); // красный
// Следующий цвет используется для 3D-элементов и панелей
wxColour color(wxSystemSettings::GetColour(wxSYS_COLOUR_3DFACE));
```

Также можно использовать указатель на wxTheColourDatabase, чтобы добавить новый цвет, найти объект класса wxColour по заданному имени, или найти имя, сопоставленное данному цвету:

```
wxTheColourDatabase->Add(wxT("PINKISH"), wxColour(234, 184, 184));
wxString name = wxTheColourDatabase->FindName(wxColour(234, 184, 184));
wxString color = wxTheColourDatabase->Find(name);
```

Чтобы определить текущие параметры обводки (пера, карандаша) для контекста устройства необходимо передать объект класса wxPen в метод SetPen. Обводкаопределяет цвет, толщину и стиль. Класс wxPen достаточно «легкий», поэтому можно определять его экземпляры в стеке вместо того, чтобы постоянно хранить его. Метод SetCap позволяет изменить способ начертания концов обводки: с установленным wxCAP_ROUND (по умолчанию) концы закругляются, с wxCAP_PROJECTING концы рисуются квадратными, а с wxCAP_BUTT концы рисуются квадратными и не один из них не отбрасывается. С помощью метода SetJoin можно определить способ соединения линий. По умолчанию используется стиль wxJOIN_ROUND, который рисует углы скругленными. Другие возможные значения — wxJOIN_BEVEL и wxJOIN_MITER. Существует несколько предопределенных обводок, которые можно использовать: wxRED_PEN, wxCYAN_PEN, wxGREEN_PEN, wxBLACK_PEN, wxWHITE_PEN, wxTRANSPARENT_PEN, wxBLACK_DASHED_PEN, wxGREY_PEN, wxMEDIUM_GREY_PEN и wxLIGHT_GREY_PEN. Эти константы являются указателями, а поэтому их надо разыменовывать при передаче в SetPen. Существует также объект wxNullPen (это именно объект, а не указатель на

него), который обозначает неинициализированную обводку. Его можно использовать, чтобы сбросить текущую обводку в контексте устройства.

Вот несколько примеров создания обводок:

```
// Сплошная красная обводка
wxPen pen(wxColour(255, 0, 0), 1, wxSOLID);
wxPen pen(wxT("RED"), 1, wxSOLID);
wxPen pen = (*wxRED_PEN);
wxPen pen(*wxRED_PEN);
```

В последних двух строчках примера используется подсчет ссылок, поэтому внутренние данные обводок указывают на данные wxRED_PEN. Одним из способов сократить число созданий/удалений классов обводок без их хранения внутри ваших классов заключается в использовании глобального указателя wxThePenList для создания и хранения необходимых вам обводок. Например, wxPen* pen = wxThePenList->FindOrCreatePen(*wxRED, 1, wxSOLID); В таком случае обводка будет хранится в wxThePenList и освобождаться при закрытии программы. Однако необходимо быть острожными и бесконтрольно не использовать данный метод, так как это может привести к излишнему расходу памяти на хранение ненужных обводок. Кроме того, необходимо учитывать нюансы механизма подсчета ссылок, которые недавно упоминались. Метод RemovePen позволяет удалить обводку из списка, без освобождения памяти. Текущая кисть, задаваемая с помощью SetBrush, определяет цвет и стиль закраски. Существует возможность задать цвет фона контекста устройства с помощью объекта wxBrush. Также как и в случае с wxPen, wxBrush обладает низкими накладными расходами на создание, а поэтому может создаваться в стеке. Существует несколько предопределенных кистей, которые вы можете использовать: wxBLUE_BRUSH, wxGREEN_BRUSH, wxWHITE_BRUSH, wxBLACK_BRUSH, wxGREY_BRUSH, wxMEDIUM_GREY_BRUSH, wxLIGHT_GREY_BRUSH, wxTRANSPARENT_BRUSH, wxCYAN_BRUSH и wxRED_BRUSH. Все эти константы являются указателями, а поэтому их

необходимо разыменовывать при передаче в SetBrush. Существует также объект wxNullBrush (это именно объект, а не указатель на него), который обозначает неинициализированную кисть. Его можно использовать, чтобы сбросить текущую кисть в контексте устройства. Вот несколько примеров создания кистей:

```
// Сплошная красная кисть
wxBrush brush(wxColour(255, 0, 0), wxSOLID);
wxBrush brush(wxT("RED"), wxSOLID);
wxBrush brush = (*wxRED_BRUSH); // дешевая операция
wxBrush brush(*wxRED_BRUSH);
```

Так же, как и в случае с wxPen, для wxBrush существует аналогичный список wxTheBrushList, который можно использовать для кеширования кистей:

```
wxBrush* brush = wxTheBrushList->FindOrCreateBrush(*wxBLUE, wxSOLID);
```

Используйте его с осторожностью, чтобы избежать излишнего распространения кистей, учитывая при этом возможные эффекты механизма подсчета ссылок.

Варианты заданий.

Вариант 1

Создать простой рисунок, используя класс wxColour.

Вариант 2

Создать простой рисунок, используя класс wxPen.

Вариант 3

Создать простой рисунок, используя класс wxBrush.

Лабораторная работа 4

Сокеты.

Задание.

Изучить приемы работы с сокетами в wxWidgets.

Справочный материал.

Адресация сокета

Все классы, представляющие собой адрес сокета, являются производными от базового класса wxSockAddress, благодаря которому в методы можно легко передать адрес, независимо от используемого протокола. Класс wxIPV4address предоставляет все необходимые методы для задания удаленного хоста, используя текущую стандартную для Интернет схему адресации IPv4.

Примечание: При представлении адреса в виде беззнакового целого числа необходимо учитывать порядок байтов в сети. Порядок байтов в сети соответствует порядку big endian (32- и 64-битные архитектуры Intel или AMD используют little endian, а архитектура Apple — big endian). В зависимости от того, как беззнаковые целые адреса хранятся или вводятся, вам, возможно, потребуется использовать макрос wxINT32_SWAP_ON_LE, который поменяет порядок байтов на правильный только для платформы little endian. Метод Hostname может принимать строку адреса в формате wxString (например, www.wxwidgets.org) или IP-адрес в виде 4-байтового целого числа (как уже отмечалось ранее в big endian). Без каких-либо параметров, Hostname возвращает имя хоста в настоящий момент. Метод Service устанавливает удаленный порт, который можно задать используя строковое (wxString) описание для указания известного порта или unsigned short для указания произвольного порта. Вызов метода без параметров возвращает номер порта, выбранный в настоящий момент. IPAddress возвращает адрес удаленного хоста в виде IP-адреса в формате wxString. AnyAddress устанавливает адрес равным любому адресу текущей машины. Эквивалентен использованию адреса INADDR_ANY.

Клиенты сокета

Класс wxSocketClient является производным от wxSocketBase и наследует все методы базового класса. Добавленные методы предназначены для инициализации и установки подключения к удаленному серверу. Метод Connect принимает аргумент класса wxSockAddress, указывающий клиенту

адрес и порт сервера для подключения. Как упоминалось ранее, вы должны использовать классы наподобие wxIPV4address, а не wxSockAddress напрямую. Второй параметр, по умолчанию равный true, указывает на то, что вызов Connect должен быть блокирующим. Если такой вызов делать в основном потоке, то на время подключения GUI будет заблокирован. Метод WaitOnConnect можно использовать после вызова Connect в случаях, если вызов Connect был деблокирующим В первом параметре передается время ожидания в секундах, а во втором — время ожидания, но в миллисекундах Если соединение успешно установлено или запрос провалился (например, было задано неправильное имя хоста), то возвращается true. Если время ожидания истекло, то возвращается false. Если параметр задать равным -1, то будет использовано значение по умолчанию, которое составляет 10 минут, но может быть переопределено с помощью SetTimeout.

События от сокетов

Все события от сокетов можно отфильтровать с помощью одного макроса —EVT_SOCKET.

EVT_SOCKET(идентификатор, функция) посылает события от сокета с определенным идентификатором указанной функции. Функция должна принимать wxSocketEvent в качестве аргумента. Хотя класс wxSocketEvent сам по себе очень простой, но содержит тип события и сокет, для которого данное событие имело место. В результате отпадает необходимость хранить указатель на сокет.

Основные методы wxSocketEvent

wxSocketEvent используется в качестве параметра для обработчиков событий сокетов. Метод GetSocket возвращает указатель на wxSocketBase, который является сокетом, породившим данное событие. GetSocketEvent возвращает тип произошедшего события.

Прием и передача данных через сокет

wxSocketBase поддерживает различные базовые и продвинутые методы для чтения и записи данных через сокет. Все операции чтения и записи сохраняют результаты операции и позволяют через LastCount получить число прочитанных байт, а также последнюю ошибку, используя LastError.

Чтение

Discard удаляет все пришедшие данные из буфера сокета. Peek позволяет получить доступ к данным внутри буфера, не удаляя их оттуда. Методу необходимо передать буфер для считывания данных, а также его размер. Read извлекает данные из буфера сокета и копирует их в переданный буфер, вплоть до указанного размера. ReadMsg читает данные, посылаемые WriteMsg в указанный буфер с определенным размером. Если буфер переполнится, то лишние данные будут отброшены. ReadMsg всегда пытается получить все сообщение, посланное с использованием WriteMsg, если не происходит ошибка. Unread копирует данные из предоставленного буфера обратно в буфер сокета. Также необходимо указать количество данных для возвращения.

Запись

Write передает данные через сокет. Вам необходимо указать передаваемые данные и количество байт для отправки. WriteMsg очень похож на Write, но при передаче также добавляет специальный заголовок. Заголовок используется для того, чтобы клиент точно знал, сколько данных необходимо прочитать. Обратите внимание, что данные, передаваемые с использованием WriteMsg, на удаленной стороне необходимо читать с помощью ReadMsg.

Создание сервера

Класс wxSocketServer добавляет к wxSocketBase несколько методов для создания слушателя и приема соединений. Чтобы создать сервер необходимо определить порт для приема входящих соединений. При конструировании wxSocketServer использует тот же класс wxIPV4address, что и для wxSocketClient, но без указания удаленного хоста. В большинстве случаев,

после создания серверного сокета необходимо вызвать метод Ok, чтобы проверить, что сокет успешно создан и принимает входящие соединения.

Основные методы wxSocketServer

wxSocketServer принимает адрес объекта с указанием порта для прослушивания, а также необязательных флагов Accept принимает входящий запрос на соединение, если таковой имеется, и создает для него сокет. Опционально можно указать необходимость ожидать новое подключение или сразу возвращать NULL, если ожидающего входящего соединения нет. Если вы укажете необходимость ожидания, то GUI на это время будет заблокирован. AcceptWith работает аналогично Accept, но ему необходимо передать ссылку на уже существующий объект класса wxSocketBase. Возвращается логическая переменная, указывающая было ли соединение принято. WaitForAccept принимает на вход количество секунд и миллисекунд, означающее время в течении которого необходимо ожидать нового соединения. Метод возвращает true, если в указанный период времени был запрос на новое соединение и false в противном случае.

Класс wxSocketServer добавляет к wxSocketBase несколько методов для создания слушателя и приема соединений. Чтобы создать сервер необходимо определить порт для приема входящих соединений. При конструировании wxSocketServer использует тот же класс wxIPV4address, что и для wxSocketClient, но без указания удаленного хоста. В большинстве случаев, после создания серверного сокета необходимо вызвать метод Ok, чтобы проверить, что сокет успешно создан и принимает входящие соединения.

Обработка установления нового подключения

Когда слушающий сокет обнаруживает новое входящее соединение, то он сообщает об этом обработчику событий. Обработчик события может принять соединение и выполнить любые необходимые действия. Предполагая, что соединение будет некоторое время использоваться и сразу не закроется, вы можете назначить обработчик событий для нового сокета.

Помните, что слушающий сокет продолжает принимать новые подключения, пока не будет закрыт, а новые сокеты создаются для каждого нового соединения. В течение жизни программы-сервера слушающий сокет может породить тысячи новых сокетов.

Список использованных источников

1. Свободная энциклопедия Википедия http://ru.wikipedia.org/wiki/WxWidgets
2. Официальный сайт wxWidgets http://www.wxwidgets.org/
3. Cross-Platform GUI Programming with wxWidgets J. Smart, K. Hock