ҚАЗАҚСТАН РЕСПУБЛИКАСЫ ҒЫЛЫМ ЖӘНЕ ЖОҒАРЫ БІЛІМ МИНИСТРЛІГІ

«Л.Н. ГУМИЛЕВ АТЫНДАҒЫ ЕУРАЗИЯ ҰЛТТЫҚ УНИВЕРСИТЕТІ» КЕАҚ

Студенттер мен жас ғалымдардың
«ǴYLYM JÁNE BILIM - 2024»
XIX Халықаралық ғылыми конференциясының
БАЯНДАМАЛАР ЖИНАҒЫ

СБОРНИК МАТЕРИАЛОВ
XIX Международной научной конференции
студентов и молодых ученых
«ǴYLYM JÁNE BILIM - 2024»

PROCEEDINGS
of the XIX International Scientific Conference
for students and young scholars
«ǴYLYM JÁNE BILIM - 2024»

2024
Астана

УДК 001
ББК 72
G99

«ǴYLYM JÁNE BILIM – 2024» студенттер мен жас ғалымдардың XIX Халықаралық ғылыми конференциясы = XIX Международная научная конференция студентов и молодых ученых «ǴYLYM JÁNE BILIM – 2024» = The XIX International Scientific Conference for students and young scholars «ǴYLYM JÁNE BILIM – 2024». – Астана: – 7478 б. - қазақша, орысша, ағылшынша.

ISBN 978-601-7697-07-5

Жинаққа студенттердің, магистранттардың, докоранттардың және жас ғалымдардың жаратылыстану-техникалық және гуманитарлық ғылымдардың өзекті мәселелері бойынша баяндамалары енгізілген.

The proceedings are the papers of students, undergraduates, doctoral students and young researchers on topical issues of natural and technical sciences and humanities.

В сборник вошли доклады студентов, магистрантов, докторантов имолодых ученых по актуальным вопросам естественно-технических и гуманитарных наук.

УДК 001
ББК 72
G99

ISBN 978-601-7697-07-5

2. Burke, R. Hybrid web recommender systems. In: The adaptive web: methods and strategies of web personalization // Springer-Verlag. 2007. P.377–408.

3. Системы поддержки принятия решений: учебник и практикум для вузов / В. Г. Халин [и др.]; под редакцией В. Г. Халина, Г. В. Черновой. — М.: Издательство Юрайт, 2024. — 494 с.

4. Tatnall, Arthur, and Stephen Burgess Experiences in building and using decision-support systems in postgraduate University Courses // Interdisciplinary Journal of Information, Knowledge and Management. Vol. 2, annual 2007. P. 33-42.

УДК 004.4'2

РАЗРАБОТКА КРОССПЛАТФОРМЕННОГО ПРИКЛАДНОГО ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ С ИСПОЛЬЗОВАНИЕМ ФРЕЙМВОРКА QT И ЯЗЫКА ПРОГРАММИРОВАНИЯ C++

Мендешев Бауыржан Бақытұлы
mendeshev.bauyrzhan@mail.ru
Студент факультета информационных технологий ЕНУ им. Л.Н.Гумилева, Астана, Казахстан
Научный руководитель – **Садвакасова К.Ж.**

Ключевые слова: кроссплатформенность, объектно-ориентированное программирование, графический интерфейс пользователя, фреймворк, объектная модель, иерархия классов, сигналы и слоты, метаобъектный компилятор.

Введение. Сегодня практически невозможно представить себе приложение, не обладающее интерфейсом пользователя. Понятие Software (программный продукт), Apps (приложения) и GUI (Graphical User Interface, графический интерфейс пользователя) неразрывно связаны друг с другом. Хотя и на сегодняшний день каждая из существующих операционных систем, или по крайней мере их подавляющее большинство обладают встроенным API, который содержит весь необходимый набор инструментов для разработки прикладного ПО, с графическим интерфейсом пользователя. Использование этих доступных инструментов требует больших затрат времени и практического опыта. Даже библиотеки, призванные облегчить процесс написания программ, не дают процессу создания программ и приложений той простоты и лёгкости, какой хотелось бы. Поэтому и сегодня разработчики по-прежнему тратят массу времени на реализацию интерфейса пользователя. Но самый большой недостаток, связанный с применением таких библиотек, - это платформозависимость [1].

Взамен платформозависимой разработке, которая предполагает создание приложений, работающих на одной операционной системе или устройстве, существует иной подход (платформонезависимая), при котором программный продукт изначально создаётся с целью использования на нескольких платформах или с возможностью беспроблемного переноса на другие платформы при наличии такой необходимости. Платформонезависимая, или как часто говорят кроссплатформенная разработка – это будущее программной индустрии. С каждым днём она будет приобретать всё более возрастающее значение. Принцип кроссплатформенности способен в разы облегчить процесс разработки приложений, так как отпадает необходимость написания разного программного кода для разных платформ. Вместе с этим применение подобного подхода в разработке значительно улучшает качество конечного программного продукта, так как приложение будет тестироваться на нескольких платформах, а возникающие ошибки исправляться в одном и том же коде программы. Как видно из выше сказанного кроссплатформенная разработка ПО обладает явными преимуществами, поскольку делает процесс разработки легче, быстрее и безопаснее, не говоря уже о том, что продукт, который не привязан к одной платформе будет иметь большее количество пользователей, что положительно скажется на коммерческом успехе продукта.

Qt, о которой будет идти дальше речь – это как раз яркий пример средства кроссплатформенной разработки. Qt – это фреймворк (библиотека классов) использующееся для разработки кроссплатформенного, пользовательского ПО с графическим интерфейсом различной направленности, начиная с музыкальных плееров и графических редакторов заканчивая приложениями для работы с базами данных. Фреймворк Qt разрабатывается и поддерживается коммерческой компанией The Qt Company и сообществом разработчиков Qt Project. Распространяется под коммерческой и открытой лицензиями. Коммерческая лицензия предоставляет полный функционал и доступ к возможностям фреймворка, тогда как открытая лицензия имеет некоторые ограничения, в большинстве касающиеся инструментов разработки для встраиваемых систем (Qt Embedded). Qt разработан на языке программирования C++ и официальная версия фреймворка поддерживает разработку именно на этом языке программирования. Но стоит отметить то что помимо официальной версии фреймворка существуют API позволяющие использовать преимущества фреймворка на других языках программирования: Python – PyQt, PySide; Ruby – QtRuby; Java – QtJambi; PHP – PHP-Qt и другие [2]. Далее в статье будет говориться именно об официальной версии фреймворка – Qt6, так как она активно обновляется и предоставляет самый полный доступ к возможностям фреймворка.

Возможности и примеры использования фреймворка. Как уже говорилось ранее, Qt это кроссплатформенный фреймворк и предоставляет поддержку большого числа операционных систем: Microsoft Windows, Mac OS X, Linux, FreeBSD и других клонов UNIX с X11, а также и для мобильных операционных систем iOS, Android, Windows Phone, Windows RT и BlackBerry. Более того, благодаря встраиваемому пакету Qt Embedded все возможности Qt доступны также и в интегрированных системах (Embedded Systems). Qt как фреймворк предоставляет широкий спектр возможностей. В первую очередь это возможности разработки графического интерфейса пользователя (GUI). Библиотека Qt содержит большое количество самых разнообразных виджетов, начиная от простых кнопок, заканчивая виджетами списков, таблиц, деревьев и многими другими. Также есть классы для работы с сетью, в частности с протоколами TCP и UDP и с HTTP запросами. Кроме этого, в Qt есть возможность работы с базами данных. Фреймворк Qt поддерживает большинство баз данных от MySQL до PostgreSQL и Oracle. Есть возможность работы с мультимедиа, проигрывания большинства современных форматов аудио и видеоданных, работы с 2D/3D графикой и использованием OpenGL и Vulkan. Иметься возможность отображение веб-страниц внутри приложения за счёт наличия встроенного веб-движка (см. Таб.1 и Таб.2). Для ознакомления с полным перечнем возможностей фреймворка можно обратиться к документации или информации на официальном сайте Qt. Кроме стандартных возможностей самого фреймворка, также имеются готовые прикладные инструменты для работы с картой и навигацией, разнообразные способы ввода информации такие как сенсорный ввод для мобильных устройств и поддержка электронного пера для графических планшетов. Также есть инструменты локализации для перевода приложения на разные языки.

На сегодняшний день Qt – это технология, которая широко применяется программистами всего мира. Существует более 4 тысяч компаний которые работаю с этим фреймворком. Вот несколько примеров компаний, работающих с Qt: Amazon, Adobe, Bosh, Amd, Canon, BMW, Disney, Cisco Systems, Intel, IBM и другие. Есть очень много программных продуктов разработанных с использованием этой технологии. Далее приведены несколько ярких показательных примеров использования данного фреймворка:

– Рабочий стол KDE Software Compilation 4**,** используемый в таких операционных системах как Linux и FreeBSD;
– Программа для работы с 3D графикойAutodesk Maya;
– Мессенджер Telegram (десктопная версия);
– Песочница Virtual Box от Sun Microsystems;
– Программа Adobe Photoshop Album для обработки растровых изображений и создания фотоальбомов;

– Карта мира Google Earth, позволяющая просматривать карту земли и других планет и их спутников;

– Программа для просмотра карты и навигации 2GIS, которая стала очень популярной в нашей стране;

– Проигрыватель аудиоданных VLC media player, начиная с версии 0.9;

– Программы официальных клиентов виртуальных валют Bitcoin и Lite coin и т.д.

Структура фреймворка. Qt – полностью объектно-ориентированная библиотека. Также Qt является расширяемым и поддерживает технику компонентного программирования. Поскольку Qt разработан на C++, и разработка программ с использованием этого фреймворка ведётся на языке программирования C++, который считается компилируемым языком низкого уровня, то о производительности и эффективности написанных программ можно не беспокоится. В фреймворке Qt для организации связи между объектами используется специальная объектная модель Qt, поддерживающая механизм сигналов и слотов о которой пойдёт речь позже. Предоставляемая система расширений (plug-ins) позволяет создавать модули, расширяющие функциональные возможности создаваемых приложений. Если говорить о документированности фреймворка, то можно сказать что Qt обладает развитой системой документаций, которая позволяет без проблем находить любую необходимую информацию в процессе разработки.

Фреймворк Qt, как уже говорилось ранее представляет из себя библиотеку классов, организованных в определённую иерархию классов. Каждый класс верхнего уровня является независимым программным модулем, отвечающий за какую-то часть функциональности фреймворка. Общее количество всех классов фреймворка Qt в общем счёте превышает 2000.

Таблица 1. Модули фреймворка Qt:

Qt Essentials	Базовые модули, которые работают на всех платформах, поддерживают бинарную совместимость для всех версий Qt5.
Qt Add-Ons	Предоставляет дополнительный, специфический для конкретной платформы функционал (Bluetooth, Wi-Fi, метки NFC и т.д.)
Value-Add Modules	Набор модулей для работы со встраиваемыми системами (Qt Embedded), доступны только под коммерческой лицензией.
Technology Preview Modules	Модули, которые находятся в разработке, но доступны для тестирования.

Таблица 2. Основные классы модуля Qt Essentials:

Qt Core	Основополагающий модуль, состоящий из базовых классов, не связанных с графическим интерфейсом. В него входят классы для работы с таймерами, потоками, STL-совместимыми контейнерами, строками, а также классы для работы с сигналами и слотами.
Qt GUI	Содержит базовые классы для графического интерфейса пользователя, включая OpenGL и Vulkan.
Qt Widgets	Содержит реализации основных элементов графического интерфейса. Дополняет модуль QtGUI «строительным материалом» для графического интерфейса в виде виджетов на C++.
Qt Multimedia	Предоставляет набор инструментов для работы с аудио, видео, радио и функциями камеры. Включает в себя как низкоуровневый функционал декодирования и кодирования видео, так и высокоуровневый функционал для воспроизведения конкретных файлов или создания плей-листов.
Qt Network	Модуль для работы с сетью. Есть поддержка всех современных протоколов передачи данных (TCP, UDP, HTTP и другие).
Qt SQL	Модуль для работы с базами данных. Есть поддержка всех современных баз данных (MySQL, PostgreSQL, Oracle и другие).

Qt QML	Модуль содержащий движок для языка QML и JavaScript.
Qt Quick	Модуль содержащий описательный фреймворк для быстрого создания графического интерфейса.

Объектная модель Qt. Фреймворк Qt спроектирован по принципу объектно-ориентированного программирования и соответственно поддерживает парадигму ООП. Объектная модель Qt подразумевает, что все механизмы технологий поддерживают ООП и построены строго на объектах. Класс QObject входящий в состав модуля QtEssentials является основным, базовым классом для большинства классов Qt. Подавляющее большинство классов Qt являются его наследниками. Класс QObject содержит в себе поддержку сигналов и слотов (signal/slot), таймера, механизма объединения объектов в иерархии, событий и механизма их фильтрации, приведения типов, и свойств объектов. Здесь стоит сделать акцент на поддержку сигналов и слотов, так как эти средства, позволяют эффективно производить обмен информацией о событиях, вырабатываемых объектами. Этот механизм является основой объектной модели Qt, Можно сказать что концепция сигналов и слотов – это нечто вроде новшества в мире разработки пользовательского ПО, которое используется тем не менее и в некоторых других библиотеках C++ [3]. До появления механизма сигналов и слотов использовалась старая концепция функций обратного вызова (callback functions), лежащая в основе X Window System. Эта концепция основана на использовании обычных функций, которые вызываются в ответ на действия пользователя. Вполне естественно то что использование функций обратного вызова серьёзно усложняет понимание исходного кода, и делает его менее понятным. Помимо этого, при таком подходе отсутствует возможность проверки типов возвращаемых значений, потому что во всех случаях функция возвращает указатель на пустой тип void.

Механизм сигналов и слотов полностью заменил старую изжившую себя модель функций обратного вызова. Он очень гибок, и полностью поддерживает ООП парадигму. Благодаря этому подходу можно организовывать связи между объектами классов. В таких связях сигнал является специальным методом класса, который отправляет сигнал, а слот методом другого класса, который принимает сигнал и выполняет определённое действие. На интуитивном уровне это очень легко понять, поскольку в реальной жизни люди сталкиваются с сигналами и слотами ежедневно. Можно привести простой житейский пример – человек сидит дома и ему стучат в дверь, он слышит это и идёт открывать дверь. На языке сигналов и слотов стук в дверь — это своеобразный «сигнал», а действие человека в ответ на стук в дверь это своеобразный «слот».

В разработке программ на Qt под сигналом имеется ввиду методы, которые способны пересылать сообщения. Сигнал может появится если возникнет сообщение об изменении состояния виджета – например, о нажатии кнопки на окне приложения. При этом соединяемые объекты могут быть абсолютно независимы и реализованы отдельно друг от друга. За реализацию сигналов и слотов, за их правильное взаимодействие и правильную работу полностью отвечает разработчик. Объект, отправляющий сигналы, может не знать, что эти сигналы принимает другой объект. Такое делегирование позволяет разбить большой проект на компоненты. Каждый компонент, разработанный отдельными разработчиками в последствии, во время конечной сборки проекта может легко быть соединён посредством сигналов и слотов. Таким образом механизм сигналов и слотов позволяет реализовать приложения по правильным архитектурным принципам, обеспечивая слабую зацепление между программными модулями проекта. Это делает библиотеку Qt особенно привлекательной для реализации компонентно-ориентированных приложений [4].

Из всего выше сказанного может показаться что механизм сигналов и слотов — это отличное средство разработки (так оно и есть), для его применения необходимо использование специального препроцессора MOC (Meta Object Compiler, метаобъектный компилятор) [5]. Дело в том, что механизм сигналов и слотов не является стандартным средством языка программирования C++. Вообще язык программирования C++ изначально не создавался для

проектирования графического интерфейса пользователя, и является плохо расширяемым языком. Проблема расширения языка C++ решается с помощью MOC компилятора. Он анализирует классы на наличие в их определении специального макроса Q_OBJECT и внедряет в отдельный файл всю необходимую дополнительную информацию. Это происходит автоматически, без непосредственного участия разработчика. Подобная операция автоматического создания кода не противоречит привычному процессу программирования C++ - ведь стандартный препроцессор перед компиляцией самой программы тоже создаёт промежуточный код, содержащий исполненные команды препроцессора.

Тема объектной модели Qt весьма обширна, а детали работы сигналов и слотов могут занимать целую главу в учебниках по Qt. В данной статье приведено лишь краткое описание важных моментов фреймворка Qt. В процессе работы над статьёй были рассмотрены вопросы кроссплатформенной разработки приложений и разработки графического интерфейса пользователя, преимущества кроссплатформенной разработки над платформозависимой разработкой. Далее в статье фреймворк Qt был предложен как альтернативное решение платформозависимой разработке на C++. В статье раскрыты основные возможности и структура фреймворка. Были рассмотрены вопросы объектно-ориентированного программирования и объектной модели Qt. По результатам проведённого анализа можно сделать вывод, что Qt с точки зрения разработки кроссплатформенного пользовательского ПО, является действительно мощной технологией и обладает большими возможностями и перспективами в программной индустрии.

Демонстрационный проект. Во время работы над статьёй, для наглядной демонстрации возможностей фреймворка Qt был разработан небольшой проект – игра «Шахматы» (см. Рис.1 и Рис.2).

Рисунок 1 – Интерфейс программы

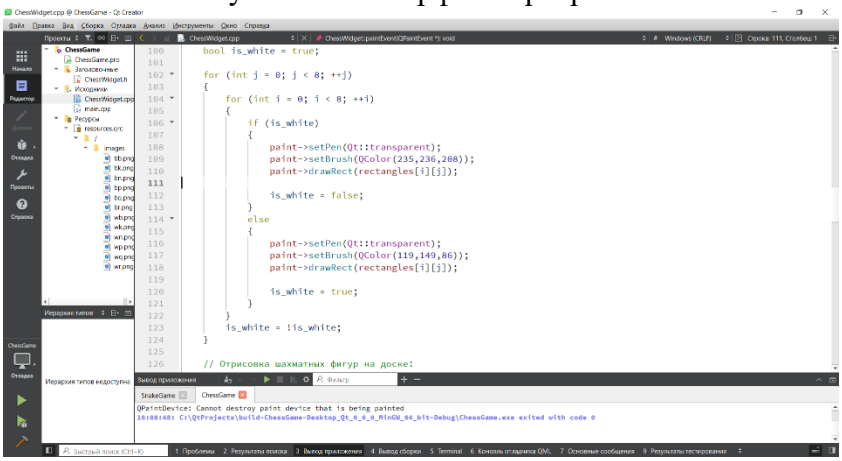

Рисунок 2 – Отрывок кода из проекта

Список использованных источников

1. Шлее М. – Qt 5.10. Профессиональное программирование на C++. — СПб: БХВ-Петербург, 2018. – 1072 с.
2. Ларионенко Р. – Курс прикладного программирования Qt Framework.
3. Стивен П. – Язык программирования C++. Лекции и упражнения, 6-е изд.: Пер. с англ. – М.: ООО «И.Д.Вильямс», 2012 – 1248с.
4. Саммерфилд М. – Qt. Профессиональное программирование. Разработка кроссплатформенных приложений на C++. — СПб: Символ-Плюс, 2011. – 560с.
5. https://ru.wikipedia.org/wiki/Qt

УДК 007.51

ИНЖЕНЕРНЫЕ РЕШЕНИЯ В DIY: МОДУЛЬНЫЙ ДЖОЙСТИК С ТЕХНОЛОГИЕЙ ДАТЧИКОВ ХОЛЛА И ДВУХСТУПЕНЧАТЫМ ТРИГГЕРОМ

Мұрат Бекежан Талғатұлы
bekezhan.murat.2021@gmail.com
Студент образовательной программы «6B06102 - Вычислительная техника и программное обеспечение», Университет «Туран», Алматы, Казахстан
Научный руководитель - Ж.А. Муканова

Современный рынок предлагает широкий выбор джойстиков для авиасимуляторов от разных производителей, таких как Logitech, Thrustmaster, FR-Tec и др. [1]. Данные джойстики ориентированы на пользователей с различным уровнем подготовки и бюджетом. Однако соотношение цены и качества, реалистичность механизмов, долговечность компонентов и ограниченные возможности кастомизации могут сделать выбор джойстика сложной задачей для пользователя. Для оценки этих показателей были проанализированы некоторые модели джойстиков на рынке. Они были разделены на три категории по цене: бюджетный сегмент - Logitech Extreme 3D Pro ($34.99 [2]), средний ценовой сегмент - Thrustmaster T16000M ($69.99 [3]) и премиум-сегмент - Thrustmaster Hotas Warthog Flight Stick ($349.99 [4]).

Целью данного исследования является разработка модульного джойстика, который может быть адаптирован к потребностям пользователей, что позволит расширить возможности персонализации и повысить уровень погружения в авиасимуляторы. Модульная конструкция джойстика позволяет ему развиваться от бюджетного варианта до премиум-сегмента.

Анализ рынка HOTAS джойстиков показывает, что бюджетные модели не обеспечивают достаточной реалистичности и погружения, а модели среднего и высокого уровня требуют значительных инвестиций. Был выявлены следующие проблем:

1) Механизм триггера. В реалистичных джойстиках используется двухступенчатый спусковой крючок, который обеспечивает более реалистичные ощущения. Этот механизм редко встречается в бюджетных моделях [5].

2) Долговечность. Джойстики низкого уровня со временем теряют точность из-за износа потенциометров [6, 7]. Датчики Холла более надежны и обеспечивают стабильность и точность измерений на длительный срок. Такой подход позволяет повысить качество джойстиков низкого уровня, что делает их сопоставимыми с моделями высокого уровня, сохраняя доступность для пользователей.

3) Модульность и кастомизация. Традиционные джойстики низкого уровня не имеют модульной конструкции, что ограничивает возможности кастомизации. Такой подход открывает потенциал для глубокой кастомизации каждой части устройства под конкретного потребителя.

Разработка джойстика велась итеративно, с поэтапным улучшением через создание прототипов и их тестирование [8]. Это позволило выявить и устранить ошибки на ранних стадиях. Печатные платы (PCB) для джойстика были разработаны в KiCad. Это решение