# ЛЕКЦИЯ 12. ИНСТРУМЕНТАЛЬНЫЕ СРЕДСТВА И МЕТОДЫ ПОСТРОЕНИЯ ИНТЕРФЕЙСА ПРОГРАММ

12.1. Интерфейс пользователя

Интерфейс пользователя, он же пользовательский интерфейс (UI — англ. user interface) — интерфейс, обеспечивающий передачу информации между пользователем-человеком и программно-аппаратными компонентами компьютерной системы (ISO/IEC/IEEE 24765-2010).

Средства:

* вывода информации из устройства к пользователю — весь доступный диапазон воздействий на организм человека (зрительных, слуховых, тактильных, обонятельных и т. д.) — экраны (дисплеи, проекторы) и лампочки, динамики, зуммеры и сирены, вибромоторы и т. д. и т. ;
* ввода информации/команд пользователем в устройство — множество всевозможных устройств для контроля состояния человека — кнопки, переключатели, потенциометры, датчики положения и движения, сервоприводы, жесты лицом и руками, даже съём мозговой активности пользователя.

По наличию тех или иных средств ввода, интерфейсы разделяются на типы — жестовый, голосовой, брэйн, и т. д., возможны смешанные варианты. Средства эти должны быть необходимыми и достаточными, быть удобными и практичными, расположенными и скомпонованными разумно и понятно, соответствовать физиологии человека, не должны приводить к негативным последствиям для организма пользователя (всё это входит в понятие эргономики).

Методы:

* набор правил, заложенных разработчиком устройства, согласно которым совокупность действий пользователя должна привести к необходимой реакции устройства и выполнения требуемой задачи — т. н. логический интерфейс
* Правила эти должны быть достаточно ясны для понимания, естественны и легки для запоминания (всё это входит в понятие юзабилити)

Увеличение в устройстве (при равной функциональности) средств ввода-вывода даёт упрощение построения методов управления и упрощение правил пользования, но зато приводит к сложности восприятия информации пользователем — интерфейс становится перегруженным. И наоборот — уменьшение средств отображения и контроля приводит к усложнению правил управления — каждый элемент несёт на себе слишком много функций. Потому проектировщики интерфейсов стараются принять компромиссное решение между этими двумя крайностями в каждом отдельном случае.

12.1.1. Безопасность пользовательских интерфейсов

Одним из основных направлений исследований в области обеспечения безопасности пользовательских интерфейсов, и, в частности, визуальных интерфейсов пользователя, является разработка моделей информационной безопасности при условии комплексного учета информационных, функциональных, психофизиологических и экологических аспектов безопасности.

Это связано, прежде всего, с включением информационного фактора в состав факторов среды систем человек-компьютер и информационным характером почти всех происходящих в области распространения ИП процессов. Наименее разработанным областям проблематики защиты информации в системе человек-компьютер (СЧК) соответствуют такие угрозы, как:

* искажение воспринимаемой пользователем информации за счет её зашумления источниками среды на рабочем месте пользователя;
* потеря или искажение воспринимаемой пользователем информации из-за физической, семантической или синтаксической несогласованности её представления пользователю;
* искажение представлений пользователя о реальном состоянии объекта управления за счет скрытых информационных воздействий и неадекватное принятие им решений в процессе решения задач в рамках СЧК.

12.1.2. Виды пользовательских интерфейсов

1. Визуальный
2. Текстовый (в частности, интерфейс командной строки)
3. Графический
   1. Оконный
   2. WIMP («window, icon, menu, pointing device» - окно, значок, меню, манипулятор)
   3. Web-ориентированный
   4. Индуктивный
   5. Масштабируемый
4. Тактильный
5. Жестовый
6. Голосовой
7. Материальный (осязательный)

12.1.3. Программный интерфейс компьютерной программы

Интерфейс пользователя компьютерного приложения включает:

* средства отображения информации, отображаемую информацию, форматы и коды;
* командные режимы, язык «пользователь — интерфейс»;
* устройства и технологии ввода данных;
* диалоги, взаимодействие и транзакции между пользователем и компьютером, обратную связь с пользователем;
* поддержку принятия решений в конкретной предметной области;
* порядок использования программы и документацию на неё.

Пользовательский интерфейс часто понимают только как внешний вид программы. Однако на деле пользователь воспринимает через него всю программу в целом, а значит, такое понимание является слишком узким.

В действительности ПИ объединяет в себе все элементы и компоненты программы, которые способны оказывать влияние на взаимодействие пользователя с программным обеспечением (ПО), это не только экран, который видит пользователь.

К этим элементам относятся:

* набор задач пользователя, которые он решает при помощи системы;
* используемая системой метафора (например, рабочий стол в MS Windows®);
* элементы управления системой;
* навигация между блоками системы;
* визуальный (и не только) дизайн экранов программы;
* средства отображения информации, отображаемая информация и форматы;
* устройства и технологии ввода данных;
* диалоги, взаимодействие и транзакции между пользователем и компьютером;
* обратная связь с пользователем;
* поддержка принятия решений в конкретной предметной области;
* порядок использования программы и документация на неё.

12.2. GUI-конструкторы и среды разработок

«Конструктор графического пользовательского интерфейса» (или «GUI-конструктор»), также известный как «GUI-редактор», является инструментарием разработки программного обеспечения, который упрощает создание графического интерфейса пользователя (GUI), позволяя разработчику упорядочить элементы интерфейса (часто называемые виджетами) используя редактор drag-and-drop WYSIWYG. Без GUI-конструктора графический интерфейс пользователя должен быть создан вручную, указывая параметры каждого элемента интерфейса в исходном коде без визуальной обратной связи до запуска программы.

Пользовательские интерфейсы обычно программируются с помощью событийно-ориентированной архитектуры, поэтому GUI-конструкторы также упрощают создание кода, управляемого событиями. Этот вспомогательный код соединяет элементы интерфейса с исходящими и входящими событиями, которые запускают функции, обеспечивающие логику работы приложения.

Список GUI-конструкторов:

* App Inventor
* C++Builder
* dBase
* Delphi
* Glade Interface Designer
* Lazarus
* Microsoft Expression Blend
* MonoDevelop
* nuBuilder
* Object Pascal
* OpenWindows
* Pencil
* Qt Creator
* Scaleform
* Ultimate++
* XForms

Примеры сред разработки с встроенными GUI-конструкторами:

* Adobe Flash Builder
* Android Studio
* Code::Blocks
* dBase
* Delphi
* Eclipse
* IntelliJ IDEA
* JDeveloper
* Lazarus
* Microsoft Access
* Microsoft Visual Studio
* MonoDevelop
* NetBeans
* PascalABC.NET
* Qt Creator
* WeeScript Automation
* WinDev
* wxDev-C++
* Xcode
* Xojo

12.3. Библиотеки элементов интерфейса для разработки ПО

В каждой оконной системе существует свой набор «родных» элементов с интерфейсом низкого уровня для работы с ними.

Одни библиотеки элементов управления — это высокоуровневые «обертки» к имеющимся стандартным виджетам, упрощающие работу с ними и расширяющие их функциональность,

Другие предоставляют свой единый программный интерфейс для программирования интерфейса пользователя в нескольких платформах или оконных системах, и с целью обеспечения кроссплатформенности приводят их к единому для всех платформ API.

Третьи предоставляют кроссплатформенные возможности за счёт собственной, платформонезависимой, реализации элементов управления.

Кроссплатформенные, на основе Java:

* SWT (Standard Widget Toolkit) — в составе проекта Eclipse
* Swing — разработка Sun, компонента Java Foundation Classes (JFC) на основе AWT
* AWT — частично устаревший Abstract Windowing Toolkit (Sun, 1995)
* JavaFX — платформа для разработки RIA на Java

Кроссплатформенные, на основе C/C++, с возможностью использования в других языках:

* GTK+
* Qt

Под Microsoft Windows:

* Windows API
* MFC (Microsoft Foundation Classes)
* WTL (Windows Template Library)
* OWL (Object Windows Library, Borland Pascal 7.0, графический интерфейс)
* VCL (Visual Components Library, Borland Delphi)
* Windows Forms
* WPF (Windows Presentation Foundation или Avalon)

12.4. Современные способы построения интерфейсов

В настоящее время любая современная компьютерная программа включает в себя визуализацию данных. Ранее компьютерные программы разрабатывались в основном для ОС Microsoft Windows, однако сейчас наблюдается тенденция использования кроссплатформенных средств для разработки ПО. В результате этого появляется возможность запуска готового программного продукта на разных ОС, включая и мобильные ОС. Кроме того, в связи с бурным распространением интернета популярным направлением разработки ПО стала разработка веб-приложений или веб-сервисов.

Веб-приложение часто является полезным дополнением к клиентской прикладной программе, и важно заметить, что разработка веб-приложений в значительной степени отличается от разработки клиентских приложений и это в свою очередь создаёт некоторые проблемы. В частности, это проблема создания универсального графического интерфейса пользователя (GUI). Чтобы клиентское приложение и веб-приложение были реализованы в едином графическом стиле, необходимо приложить достаточно усилий, как разработчику интерфейса клиентского приложения, так и разработчику интерфейса веб-приложения. В конечном счёте величина усилий одного или другого разработчика будет зависеть от того, интерфейс какого приложения будет задавать общий стиль.

Рассмотрим наиболее популярные в настоящий момент способы построения интерфейсов клиентских приложений на языке C++, как наиболее используемом для разработки ПО для ОС Microsoft Windows (MS Windows) и ОС Linux. Главным средством разработки ПО для MS Windows является MS Visual Studio.

Эта интегрированная среда разработки (IDE) позволяет разрабатывать ПО на разных языках программирования, но основными языками, конечно, являются C++ и C#. Для разработки интерфейса приложения имеются два основных средства — Windows Forms (WinForms) и Windows Presentation Foundation (WPF). Большая часть существующих приложений для MS Windows разработана с использованием WinForms, однако с появлением более современных версий ОС (MS Windows 7, 8), система WPF становится более популярной.

Кроме того, последние версии MS Visual Studio позволяют использовать язык разметки HTML5 для построения интерфейсов, близких по стилю к нативным веб-приложениям. Однако стоит заметить, что коммерческая лицензия MS Visual Studio является платной, как и лицензия MS Windows, что несомненно является недостатком для низкобюджетных проектах.

Если говорить о низкобюджетных проектах, то тут наиболее подходящим вариантом является ОС Linux. Помимо того, что большинство дистрибутивов этой ОС являются абсолютно бесплатными, в том числе и для коммерческого использования, также имеется ряд бесплатных средств для разработки качественного ПО для ОС Linux.

Самым распространённым средством для разработки ПО на языке С++ является кроссплатформенный инструментарий Qt. Важно подчеркнуть, что Qt позволяет разрабатывать приложения не только для ОС Linux, но и для MS Windows, Mac OS X, Android и других UNIX-подобных ОС. Разработчики Qt предлагают как бесплатную для коммерческого использования, так и платную лицензию с дополнительными возможностями. Но исходя из современной практики разработки ПО с помощью этого инструментария, бесплатной лицензии оказывается больше чем достаточно.

Если проводить аналогию с MS Visual Studio, то в Qt мы имеем IDE Qt Creator. Здесь альтернативой WinForms являются так называемые виджеты (Qt Widgets), а альтернатива для WPF — Qt Quick. Также в Qt Creator имеется возможность создания интерфейсов на основе HTML5. Но наиболее интересным модулем инструментария является встраиваемый веб-движок WebKit, который лежит в основе всех современных веб-браузеров. Подобный модуль имеется и в MS Visual Studio, но он имеет ряд ограничений, и тем более нас больше интересуют низкобюджетные средства, которые позволяют уменьшить издержки при создании программного продукта.

Веб-движок — это ядро браузера, он отвечает за правильное отображения веб-страниц. Модуль Qt WebKit позволяет создавать интерфейс клиентского приложения с использованием техники разработки интерфейсов веб-приложений. В основе создания интерфейса веб-приложения лежит устоявшийся стек технологий. Он включает язык разметки HTML (HTML 4, 5), каскадные таблицы стилей (CSS 2, 3) и скриптовый язык JavaScript с богатым выбором дополнительных библиотек (каркасов). Отдельного внимания заслуживает тот факт, что скорость появления новых полезных каркасов для языка JavaScript стремительно растёт, а это делает разработку, насыщенных функционалом приложений, более быстрой и удобной.

12.4.1. Традиционный способ: Qt WebKit + Qt-костыли

Рассмотрим традиционный способ создания универсального GUI с помощью модуля Qt WebKit на примере модуля визуализации данных системы акустического мониторинга. Под системой акустического мониторинга подразумевается система, включающая аппаратную и программную части. Аппаратная часть системы состоит из сенсорной сети акустических датчиков, данные с которых обрабатываются на микроконтроллере и отправляются по какому-либо интерфейсу (UART, IEEE 802.X и др.) на персональный компьютер (ПК). Программная часть состоит из набора прикладных программ, работающих как на локальном ПК (клиентское ПО), так и на удалённом сервере (серверное ПО).

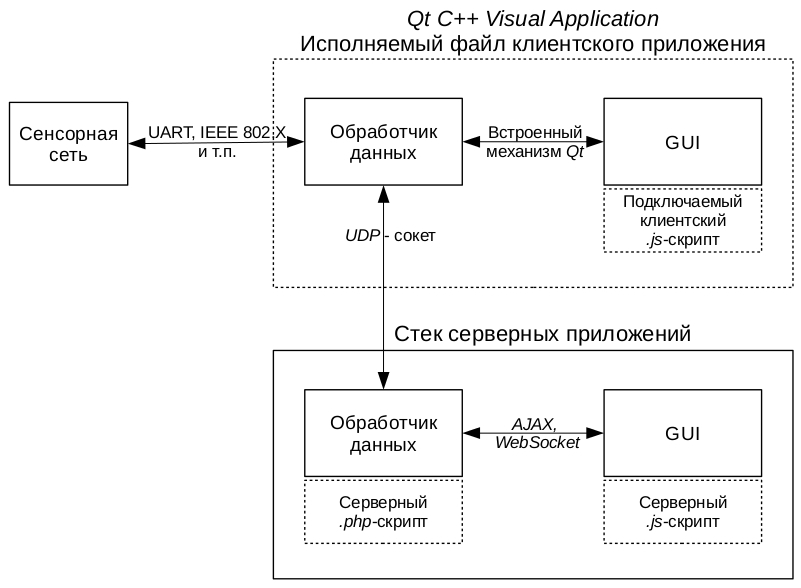


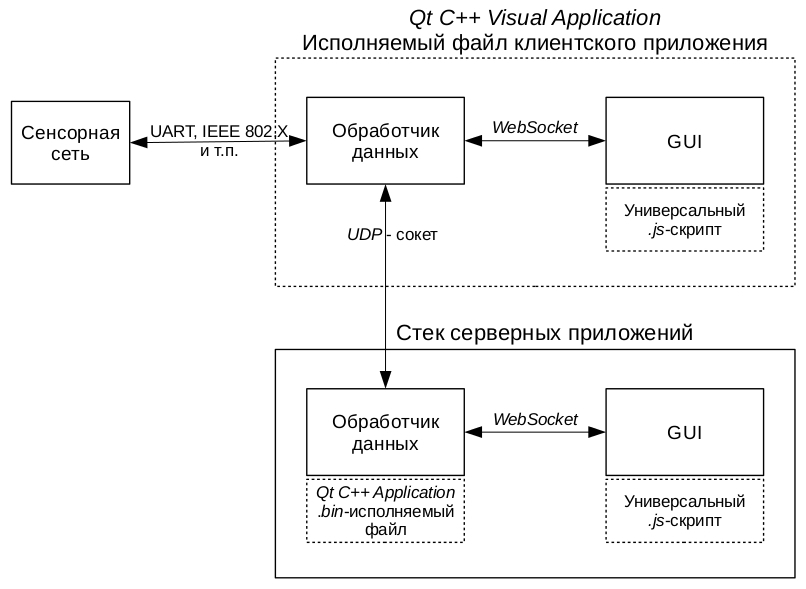
Рис. 12.1. Традиционный метод реализации универсального GUI

Традиционный метод подразумевает использование межпроцессного взаимодействия по средствам встроенного механизма Qt. Здесь подразумевается взаимодействие между основной логикой клиентского приложения, изображённой на Рис.12.1 как *Обработчик данных*, и *GUI*-элементом. Одним из недостатков такого подхода является то, что код для реализации GUI на языке JavaScript будет иметь специфические функции, которые будут актуальны только для клиентского Qt-приложения.

Для серверного приложения, отвечающего за GUI, нужен будет другой, специфичный для серверной реализации, код. Например, в случае использования PHP-скрипта для реализации основной логики серверного приложения, понадобится реализация межпроцессного взаимодействия с помощью какой-либо другой технологии (AJAX или WebSocket). Отсюда следует ещё один недостаток, а именно использование дополнительного языка программирования для реализации основной логики серверного приложения и разработка нового алгоритма межпроцессного взаимодействия.

12.4.2. Подход: Qt WebKit + WebSocket

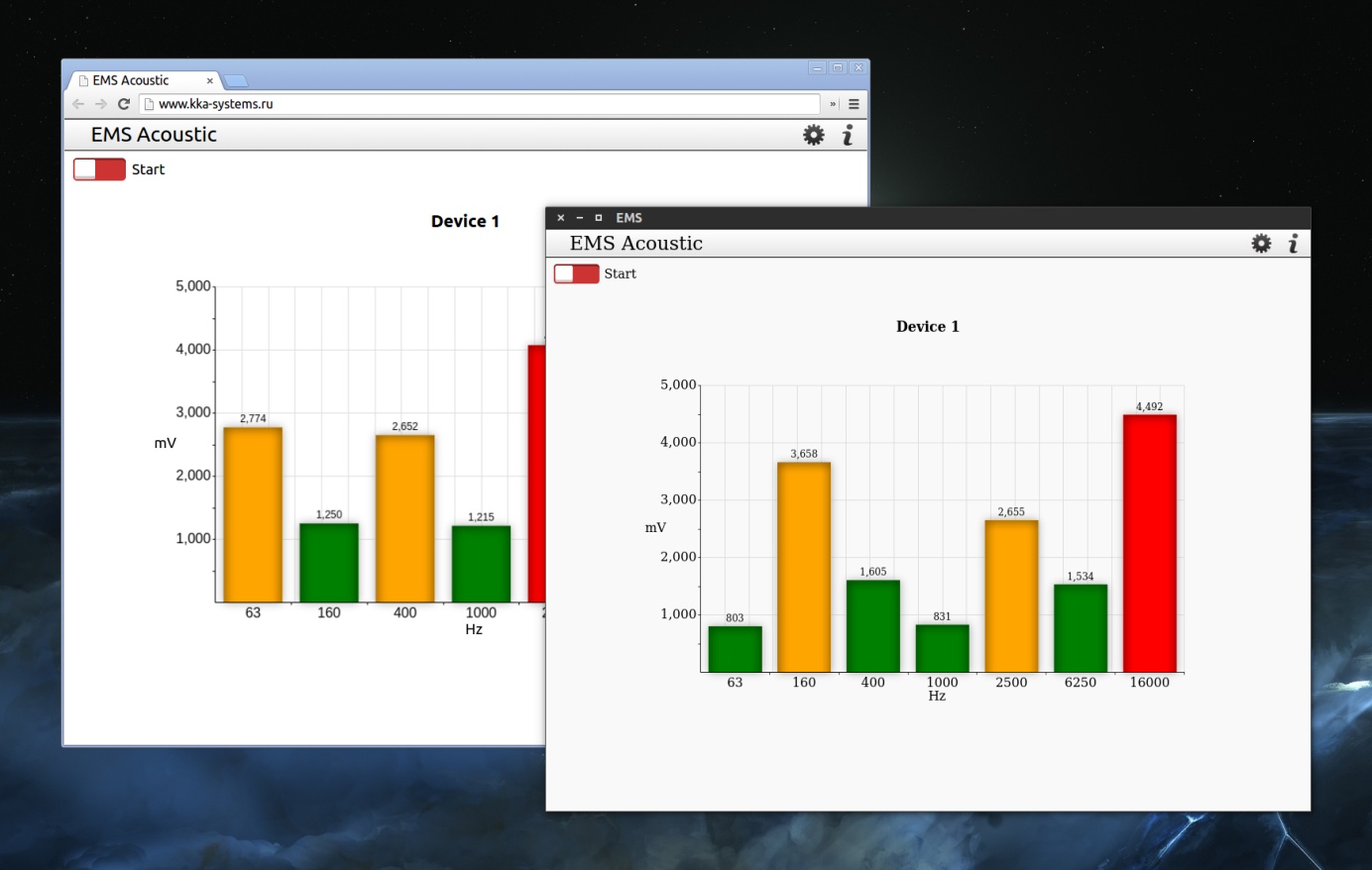
Для решения этих проблем предлагается новый метод, основанный на использования упомянутой выше технологии WebSocket. Суть метода заключается в том, чтобы унифицировать метод межпроцессного взаимодействия между основной логикой приложения и GUI, как на клиентской стороне, так и на серверной. В этом случае появляется возможность использования JavaScript кода для реализации универсального для обеих сторон GUI.

  
Рис. 12.2. Новый метод реализации универсального GUI

На Рис. 12.2. видно, что теперь для межпроцессного взаимодействия, как для клиентской, так и для серверной части используется технология WebSocket. То есть теперь мы имеем один универсальный JavaScript код для разных приложений. В этом случае необходимым условием является серверное приложение, основная логика которого реализована с помощью Qt, на не совсем привычном для веб-разработчиков, языке C++. С одной стороны такой подход к реализации серверного приложения усложняет задачу для узкоспециализированного веб-разработчика. Но с другой стороны мы имеем универсальные части кода, которые позволяют нам сэкономить время на дублировании одних и тех по смыслу алгоритмов на разных языках.

Важно также подчеркнуть, что для использования технологии WebSocket необходима дополнительная библиотека, которая имеется в интернете в свободном доступе или включается по умолчанию в более поздние версии Qt.

На Рис. 12.3 приведён пример реализации нового метода создания универсального GUI для ОС Linux. Как видно на рисунке, в конечном итоге мы получаем универсальный интерфейс, как для локального приложения, запущенного в качестве исполняемого файла ОС, так и для серверного приложения, запущенного в современном веб-браузере. Так как для разработки ПО используются кроссплатформенные инструменты, это позволяет говорить о простой переносимости программного продукта на другие ОС в будущем.

  
Рис. 12.3. Локальное (справа) и серверное (слева) приложения, запущенные на ОС Linux

12.5. Microsoft Expression Blend

Появление языка описания пользовательских интерфейсов XAML (произносится – зáммель) и новой среды разработки Microsoft Expression Blend позволяет заметно ускорить и облегчить проектирование и построение пользовательских интерфейсов как для веб-, так и для настольных приложений.

Данный язык позволяет описывать внешний вид и поведение интерфейсных элементов, устанавливать взаимодействие этих элементов с различными данными и событиями. Допускает прямое подключение к Common Language Runtime (CLR), что обеспечивает большую гибкость при проектировании ПО.

XAML – это скриптовый язык, базирующийся на XML, он имеет набор правил, которые устанавливают взаимодействие между объектами и классами, атрибутами и свойствами или событиями и пространствами имен XML и CLR. Для описания элементов, панелей, свойств текста, векторной графики и т.п. используются теги.

<Button VerticalAlignment="Top" Width="Auto" Height="Auto" Content="Button"/>

Каждый тег в XAML имеет соответствующий класс в WPF, который имеет набор инструкций, как выполнить этот тег. XAML включает в себя: панели, элементы управления, элементы управления документами и элементы векторной графики.

При создании проекта в Expression Blend каждый файл на XAML имеет файл-соратник (code-behind) на C# или VB.

XAML взаимодействует с кодом на C# или VB посредством обработчика событий, который прописывается внутри тега объекта.

Код на XAML:

<Button Content="Button" Click="Button\_Click"/>

Пример обработчика события Button\_Click на C#:

private void Button\_Click(object sender, System.Windows.RoutedEventArgs e)  
{  
MessageWindow MessageWindow = new MessageWindow();  
MessageWindow.ShowDialog();  
}

Blend имеет современный интерфейс, привычный как дизайнерам графикам, так и веб-дизайнерам. Рабочее пространство разделено на три основные части (Рис.12.4):

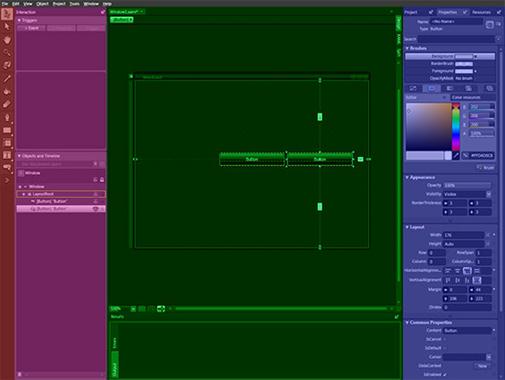


Рис. 12.4 - Рабочее пространство Microsoft Expression Blend

Панель инструментов (подкрашена красным), панели Interaction и Objects and Timeline (пурпурным) слева, основное рабочее пространство с панелью инструментов и вкладками переключения вида Design, XAML или Split посередине и панель Results в центре снизу (подкрашено зеленым) и панели Project, Properties, Resourses и Data справа (синие).

Инструменты для построения и редактирования векторной графики типичны для многих векторных редакторов и включают в себя редактор кривых, представленный инструментами: Перо (Pen), Карандаш (Pencil), инструмент выделения (Selection) и инструмент непосредственного выделения (Direct Selection), а также инструментами для построения простых геометрических форм: Прямоугольник (Rectangle), Овал (Ellipse) и Линия (Line).

Blend позволяет:

* создавать составные векторные объекты (Compound paths);
* создавать векторные объекты посредством логических операций (вычитание, сложение и т.д.) (Combining paths);
* переводить шрифт в векторный объект (Convert to Path);
* кадрировать как растровое, так и векторное изображение (Clipping paths);
* создавать маски прозрачности (Opacity masks).

Библиотека интерфейсных элементов содержит все типы стандартных интерфейсных элементов, специфические элементы Blend и элементы, содержащиеся в стиле SimpleStyles.