Санкт-Петербургский государственный университет

Математическое обеспечение и администрирование информационных систем

Системное программирование

Малиновский Илья Константинович

Проектирование и реализация расширения графического протокола для платформы UbiqMobile

Выпускная квалификационная работа

Научный руководитель:
д. ф.-м. н., профессор Терехов А. Н.

Научный консультант:
ведущий разработчик ООО "НМТ — Новые Мобильные Технологии" Невоструев К. Н.

Рецензент:
директор ООО "НМТ — Новые Мобильные Технологии" Оносовский В. В.

Санкт-Петербург
2017

SAINT-PETERSBURG STATE UNIVERSITY

Software and Administration of Information Systems

Software Engineering

Ilia Malinovskii

Design and implementation of graphic protocol extension for UbiqMobile platform

Graduation Project

Scientific supervisor:
Professor Andrey Terekhov

Consultant:
Senior Developer at New Mobile Technologies LLC Konstantin Nevostruev

Reviewer:
CEO at New Mobile Technologies LLC Valentin Onosovsky

Saint-Petersburg
2017

Оглавление

Введение

В наше время существует целый ряд популярных мобильных платформ: в их число входят Android, iOS и Windows Phone. При разработке мобильных приложений требуется учитывать особенности целевой операционной системы. Программа, созданная для одной платформы, требует значительных изменений для того, чтобы корректно исполняться в другой. Тем не менее, разработчики мобильных приложений заинтересованы в том, чтобы пользователи всех популярных платформ могли воспользоваться их продутом. Таким образом, актуален вопрос о разработке кроссплаформенных мобильных приложений, т.е. таких, которые могут работать более чем на одной операционной системе.

В основе ряда популярных систем разработки кроссплаформенных мобильных приложений лежит следующий общий принцип: программа создается на некотором высокоуровневом языке, после чего независимо транслируется в исполняемый код для каждой из поддерживаемых платформ. Среди таких систем можно выделить Xamarin (разработка ведется на языке C#), PhoneGap (используемый язык — JavaScript), Qt (основной язык программирования — C++).

С 2008 года при участии студентов и сотрудников СПбГУ ведется работа над системой Ubiq Mobile, предназначенной для создания кроссплаформенных мобильных приложений. Центральной частью системы является сервер, на котором размещается код приложения. Для работы с приложением мобильное устройство подключается к серверу при помощи предварительно установленного тонкого клиента. Принципиальной особенностью системы Ubiq Mobile является возможность сохранения состояния пользовательских сессий при разрывах мобильных соединений. На сегодняшний день приложения, созданные в системе Ubiq Mobile, могут быть запущены на устройствах, работающих под управлением Android, iOS, Windows Phone и Java ME.

Центральной составляющей системы разработки кроссплаформенных мобильных приложений является графический протокол — универсальный платформонезависимый способ описания графического ин-

терфейса приложения. Чем более гибкие средства предоставляет протокол, тем большее число возможностей каждой из поддерживаемых платформ может быть использовано в разрабатываемом приложении.

Для описания графического интерфейса приложений система Ubiq Mobile использует оригинальный древовидный бинарный протокол, надстроенный над стеком TCP/IP. Возможности предшествовавшей данной работе версии протокола ограничены фиксированным набором стандартных графических управляющих элементов (controls) и их свойств [1]. Для добавления нового элемента интерфейса требуется вносить изменения не только в код сервера, но и в каждый из тонких клиентов, что замедляет процесс разработки. Кроме того, если речь идет о специфическом управляющем элементе, который требуется только одному конкретному приложению, то его включение в протокол нецелесообразно: поддержка элемента, нужного единственному приложению, приведет к увеличению размеров серверного и клиентских компонентов системы и может замедлить их работу.

Недостаточная гибкость делает систему менее привлекательной для потенциальных пользователей — разработчиков кроссплатформенных мобильных приложений. Это препятствие можно преодолеть, сохранив общую структуру системы и внеся существенные изменения только в часть, отвечающую за передачу информации о графических элементах и их отображение.

1. Постановка задачи

Целью данной работы является расширение возможностей графического протокола, используемого в системе разработки кроссплатформенных мобильных приложений Ubiq Mobile.

В рамках данной работы были поставлены следующие задачи:

- изучив возможности технологии Windows Presentation Foundation, предложить расширение существующего протокола;
- реализовать поддержку основных возможностей расширенного протокола для эмулятора Ubiq Mobile;
- проверить работу эмулятора на тестовом приложении, использующем элементы расширенной версии протокола.

2. Обзор

2.1. Описание графических интерфейсов в системах кроссплаформенной разработки

2.1.1. Технология Xamarin

Данная технология [2] предназначена для кроссплатформенной разработки мобильных приложений для платформ iOS, Android и Windows Phone. Языком разработки является C#. Работу над фреймворком ведет с 2011 года компания Xamarin Ltd.

Отличительной особенностью Xamarin является то, что несмотря на не зависящие от целевых платформ средства для работы с хранилищами данных и унифицированный способ описания бизнес-логики приложения и взаимодействия с удаленными сервисами, для разработки графического интерфейса предполагается использовать нативные средства. Таким образом, достигается высокая производительность, но не удается добиться полноценной кроссплаформенности. Сместить баланс в сторону кроссплаформенности можно, если вместо библиотек Xamarin.Android и Xamarin.iOS использовать Xamarin.Mobile, предоставляющую программный интерфейс для общих функций систем (например, для работы с камерой или геолокацией).

Основой библиотеки компонентов графического интерфейса служат стандартные компоненты каждой из поддерживаемых мобильных операционных систем. Тем не менее, поддерживается создание пользовательских компонентов. Более того, существует магазин сторонних компонентов, в который попадают созданные как разработчиками Xamarin, так и сторонними пользователями фреймворка элементы графического интерфейса. Они могут быть реализованы как для всех поддерживаемых платформ, так и для одной конкретной.

2.1.2. Технология PhoneGap

PhoneGap — фреймворк [3] с открытым кодом для создания кроссплатформенных мобильных приложений, разрабатываемый компанией Nitobi Software с 2005 года. При создании приложений используются веб-технологии (JavaScript, HTML5, CSS3), то есть пользователь PhoneGap может разрабатывать приложения под любую из более чем десяти поддерживаемых мобильных платформ без необходимости написания кода на совместимых с этими платформами языках. Готовое приложение преобразуется к набору установочных пакетов для каждой из поддерживаемых платформ.

Основой фреймворка служат плагины — специальные пакеты, позволяющие автоматически внедрять нативный код в приложение. В частности, все базовые средства, предоставляемые фреймворком, реализованы при помощи плагинов, предоставляющих доступ к специфическим функциям платформы или устройства, которых нет у веб-приложения. Предусмотрена возможность создания пользовательских плагинов: в таком случае требуется реализовать новый компонент на JavaScript для расширения фреймворка и хотя бы для одной из поддерживаемых систем на нативном для нее языке.

2.1.3. Библиотека Qt

Qt — инструментарий [4] кроссплатформенной разработки ПО на языке программирования C++, в разные годы создававшийся и совершенствовавшийся компаниями Trolltech, Nokia, Qt Project, Digia и The Qt Company. Он позволяет исполнять написанное с его помощью программное обеспечение в большинстве современных операционных систем путем простой компиляции программы для каждой ОС без изменения исходного кода.

В Qt есть около пятидесяти готовых классов графических элементов доступных для использования. В ходе компиляции программы для определенной платформы каждый элемент заменяется соответствующим нативным аналогом.

Qt позволяет создавать новые компоненты на основе имеющихся. Такие компоненты, будучи однажды определенными, в дальнейшем могут быть переиспользованы в рамках того же проекта. Для окружения компонент, созданный на основе более простых, представляет собой "черный ящик": взаимодействие с ним происходит посредством сигналов, вырабатывающихся в ответ на определенное событие, и слотов (функций, вызывающихся при выработке соответствующего сигнала), тогда как непосредственно обратиться к составляющим частям нового компонента снаружи невозможно. Пользовательские компоненты создаются путем комбинации таких стандартных компонентов, как изображения (Image), текст (Text), форматируемое (Text Edit) и неформатируемое (Text Input) поля ввода, прямоугольник (Rectangle) и граница (Border Image). Взаимное расположение этих компонентов определяется при помощи декларативного языка QML или с использованием специального графического редактора Qt Quick Designer, встроенного в среду разработки Qt Creator.

2.2. Язык разметки XAML в системе WPF

Декларативный язык разметки Extensible Application Markup Language (XAML) [5], основанный на XML, был разработан корпорацией Microsoft для использования в различных компонентах .Net Framework 3.0. В частности, он применяется в системе построения клиентских приложений Windows Presentation Foundation (WPF) для разметки пользовательского интерфейса, поддержки событий и привязки данных.

Одной из характерных особенностей WPF, реализуемых через XAML, являются шаблоны управляющих элементов. Они позволяют менять как графическое оформление элементов, так и их структуру. Шаблон естественным образом встраивается в древовидную структуру XAML-документа в качестве значения свойства Template у изменяемого элемента.

Механизм привязки данных предоставляет возможность устанавливать зависимости между различными данными, использующимися в

приложении. Изменение одного из элементов зависимости при выполнении определенных условий приводит к изменению другого (или других) элементов. Существует возможность настройки типа зависимости (односторонняя или двухсторонняя) и события, приводящего к срабатыванию зависимости (например, сразу после изменения одного из элементов или после определенного действия пользователя), поддерживаются множественные привязки (зависимости, в которых участвует больше двух элементов) и привязки с приоритетами (упорядочивание списка привязок, относящихся к определенному элементу, и использование привязки с наивысшим приоритетом из числа успешно вернувших значение). Кроме того, специальный класс привязок (TemplateBinding) используется для установления соответствия между значениями свойств элементов внутри шаблона и значениями свойств элемента, в которому применен этот шаблон.

Стили позволяют применить одинаковое оформление сразу к нескольким элементам, описав его один раз. Тем самым, исключаются повторы, а содержимое основного файла разметки, избавленного от мельчайших деталей в описании элементов, сводится к заданию структуры интерфейса.

2.3. Текущая рабочая версия графического протокола системы Ubiq Mobile

В системе UbiqMobile графическая информация задается древовидной структурой, передающейся с сервера на установленный на мобильном устройстве тонкий клиент в виде последовательности байт. Связь между этими двумя уровнями представления данных (линейной последовательностью байт и деревом) устроена следующим образом: последовательность байт кодирует вложенные друг в друга секции. Описание каждой секции начинается с длины (включающей длину всех вложенных подсекций) и однобайтового кода (определяющего тип секции). Далее следует описание свойств секции и содержимого вложенных подсекций.

Порядок следования свойств секции определен в протоколе и не может быть изменен. Описание подсекций следует после описания всех свойств. Некоторые специфические свойства могут быть опущены, в таком случае обнуляются соответствующие этим свойствам биты в обязательных полях-флагах.

Протокол включает ограниченное количество элементов и не может быть расширен без внесения изменений как в код ядра сервера, так и в мобильные клиенты для каждой из поддерживаемых систем. Большую часть этих элементов можно отнести к одной из следующих групп: панели (например, элемент Absolute Layout, положение дочерних узлов внутри которого определяется абсолютными координатами, элемент Linear Layout, укладывающий дочерние узлы в один ряд по вертикали или горизонтали, или элемент Grid, располагающий дочерние узлы в ячейках таблицы), элементы, обрабатывающие нажатие на экран (кнопка, кнопка с изображением и специальная секция Selectable, делающая интерактивным любой элемент, содержащийся в ней), поля ввода (однострочное и многострочное текстовые поля), списочные элементы (обычный и выпадающий списки) и простые геометрические формы (отрезок, прямоугольник, эллипс и многоугольник). Помимо этих пяти групп элементов, в графический протокол входят элементы, представляющие неизменяемые текстовые поля и изображения.

Внешний вид каждого из элементов определен в протоколе и может быть подвергнут лишь незначительным изменениям посредством изменения значений отдельных свойств (например, может быть изменен цвет фона). В то же время нельзя, например, изменить форму кнопки или поля ввода или сделать элементами выпадающего списка изображения вместо текста.

3. Расширенная версия протокола

3.1. Основные нововведения

В новой версии протокола был полностью изменен подход к заданию свойств управляющих элементов. Если раньше для каждого элемента протокол определял фиксированный набор свойств и порядок их следования в описании элемента, а отсутствие определенного свойства обозначалось нулевым битом в соответствующей позиции в отвечающей за наличие свойств битовой маске, то в расширенной версии протокола маски отсутствуют в описании элементов, а каждое свойство задается как подсекция: заголовок подсекции определяет тип свойства, а содержимое — значение этого свойства. Благодаря этому свойства могут описываться в произвольном порядке. Кроме того, это нововведение позволило использовать в протоколе вложенные свойства, то есть свойства, фактически определенные в родительском элементе, для которых каждый из дочерних элементов может задавать уникальные значения.

В протокол были добавлены связки — структуры, которые позволяют привязывать значение одного параметра к значению другого. Существует два типа таких структур: простые связки и связки с данными.

Расширенная версия протокола поддерживает механизм шаблонов. Шаблон — это свойство некоторого управляющего элемента, значение которого содержит составленное из других управляющих элементов дерево. Шаблоны используются в двух случаях: для переопределения внешнего вида элемента протокола и для задания единого вида всех дочерних элементов списка.

3.2. Управляющие элементы

Элементы в расширенной версии протокола подразделены на четыре типа: примитивные, простые, шаблонизированные и пользовательские.

Элементы первого типа отличает то, что они не используются сами по себе в дереве управляющих элементов, а лишь входят в него как блоки для построения более сложных элементов. К первому типу отно-

сятся два новых элемента — Content Presenter и Items Presenter. Content Presenter представляет собой поле, данные в которое могут быть или введены пользователем приложения, или взяты из привязанного к элементу источника данных (как из внешнего, например, из базы данных, так и из внутреннего — значения выбранного атрибута любого управляющего элемента). В отличие от элемента Text View (обычного поля ввода) у Content Presenter нет рамки и строки-подсказки, что дает более гибкие возможности при использовании Content Presenter для конструирования более сложных пользовательских управляющих элементов. Items Presenter позволяет применить множество полученных из источника данных значений к однотипным элементам, внешний вид которых задан определенным в Items Presenter шаблоном. Этими элементами могут быть, например, дочерние узлы какой-либо панели. Каждый элемент источника может содержать несколько значений. В таком случае они применяются к различным свойствам соответствующего элемента списка, причем соответствие между позицией значения в источнике и свойством элемента списка задается при помощи связки с данными.

Управляющие элементы, которые нельзя составить из примитивов, относятся к простым. Это уже входящие в текущую версию протокола рамки, генерирующие события по нажатию интерактивные контейнеры, надписи, изображения и геометрические формы. Важным расширением последней группы простых элементов является добавление кривых Безье, позволяющих изображать более сложные фигуры, чем это было возможно ранее, и путей — комбинации прямых линий, квадратичных и кубических кривых Безье и команды замыкания контура. Использование замкнутых путей позволяет ко всему многообразию контуров, которые можно изобразить при помощи группы кривых, применить заливку.

За каждым из шаблонизированных управляющих элементов закреплена определенная функция, тогда как их внешний вид может быть изменен пользователем протокола. Для каждого, кроме одного, задано представление по умолчанию. Большую часть шаблонизированных элементов можно подразделить на пять групп — списочные элементы,

панели, элементы с целочисленным параметром, поля ввода и кнопки.

К списочным элементам относятся List View и Combobox. Их объединяет задание содержимого при помощи Items Presenter и возможность отметить какой-то элемент списка как выбранный. При этом в первом случае одновременно отображается фиксированное число элементов (по умолчанию — все), при необходимости (если элементов больше этого заданного числа или они не влезают на экран) используется полоса прокрутки. В свою очередь, Combobox постоянно отображает текущий выбранный элемент списка (или значение по умолчанию, если ни один элемент не выбран) и содержит кнопку, по нажатию раскрывающую аналогичный List View список в панели, с одной из сторон пристыкованной к полю для отображения элемента в фокусе. По сравнению с текущей рабочей версией протокола Combobox (Dropbox в текущей версии) позволяет — благодаря шаблонизированности — хранить не только строки, но и элементы любой структуры. Кроме того, оба управляющих элемента теперь могут привязываться к внешнему источнику данных благодаря использованию Items Presenter.

Панели используют вложенные свойства для задания положения дочерних элементов. Расширенный протокол включает в себя следующие панели:

- Stack Panel: дочерние элементы упорядочены в одну строку или в один столбец. Аналог из текущей версии протокола — Linear Layout. Использует Items Presenter для наполнения.

- Overlay Panel: дочерние элементы располагаются друг над другом. Порядок размещения элементов определяется заданным для каждого дочернего элемента панели вложенным свойством — z-координатой. Такая панель находит применение в том случае, если, например, в разных состояниях в одной и той же части экрана должно отображаться либо редактируемое поле ввода, либо надпись-подсказка. В текущей версии протокола аналогом является Absolute Layout.

- Grid Panel: дочерние элементы помещаются в клетках прямоуголь-

ной таблицы. Описание таблицы по аналогии с WPF включает число строк и столбцов, параметры каждой строки и столбца, а также задание дочерних элементов с указанием координат клеток таблицы, которые займет каждый из них [5], с. 125-137. В последнем случае используется механизм вложенных свойств: описание каждого дочернего элемента панели содержит координаты левого верхнего угла прямоугольника ячеек, занимаемого элементом, и размеры этого прямоугольника, выраженные количеством ячеек по горизонтали и вертикали. Такой подход обеспечивает краткость описания таблиц с большим числом пустых клеток, что выгодно отличает новую табличную панель в сравнении с включенной в текущую версию протокола (в последней требуется описывать содержимое каждой клетки, даже пустой). Благодаря возможности объединять соседние клетки Grid Panel (также отсутствующей в текущей версии протокола), а также задавать собственные размеры для каждой строки и стобца, этот элемент позволяет реализовать широкий диапазон способов размещения дочерних элементов панели на экране.

- Dock Panel: дочерние элементы прикрепляются к одной из четырех сторон (заданной для каждого элемента при помощи вложенного свойства) текущей доступной области панели. Свободный прямоугольник в средней части Dock Panel опционально может быть заполнен последним дочерним элементом. Эта и следующие три панели отсутствуют в текущей версии протокола.

- Expansion Panel: состоит из постоянно отображаемого заголовка, который может содержать элемент любого типа и обязательно включает в себя кнопку для раскрытия панели, и дочернего элемента, показываемого, только когда панель раскрыта.

- Tab Panel: в каждый момент отображает ровно один из нескольких дочерних элементов, позволяет переключаться между ними при помощи вкладок, расположенных на одной из боковых сто-

рон панели.

- Multiview Panel: в каждый момент отображает ровно один из нескольких дочерних элементов, позволяет переключаться между ними при помощи кнопок "вперед" и "назад", реализованных элементом Stepper.

- Slide Panel: включает два элемента типа "панель" — первый, постоянно отображающийся на всю доступную площадь, и второй, скрытый по умолчанию, выдвигающийся по нажатию кнопки или специальному жесту пользователя и частично или полностью (в зависимости от степени выдвинутости) перекрывающий первый. Единственная панель, перенесенная из текущей версии протокола практически без изменений. Отличает ее только использование двух примитивов типа Content Presenter вместо двух дочерних узлов дерева и измененный набор свойств (например, появилось свойство StyleID).

Элементы с целочисленным параметром отсутствуют как отдельные управляющие элементы в текущей рабочей версии протокола. В новой версии для каждого из элементов этого типа определен фиксированный набор компонентов. Для компонентов задается поведение, которое не может быть изменено, и стандартный внешний вид, который может быть переопределен при помощи механизма шаблонов. Обязательным свойством входящего в переопределенный шаблон компонента является тег: именно теги задают поведение, которого должен придерживаться данный компонент как часть элемента с целочисленным параметром. Эта группа шаблонизированных элементов включает в себя:

- Switch: переключатель с двумя положениями (включен/выключен), состоящий из двух компонентов: статичной шкалы с двумя позициями, между которыми может перемещаться ползунок, и, собственно, ползунка. Передвижение ползунка задается анимацией.

- Counter: абстрактный элемент (другими словами, такой элемент, экземляр которого нельзя создать в приложении), поддерживаю-

щий текущее значение, границы, в которых оно может изменяться, и шаг изменения. На его основе реализуются нижеперечисленные элементы-наследники (которые уже могут быть включены в дерево управляющих элементов):

– Progress Bar: элемент, значение которого может изменяться только под влиянием внешнего фактора, то есть источника данных. Отношение разности текущего значения и минимального к общему диапазону значений определяет заполненность шкалы. Состоит из двух компонентов: статичного фона (внешнего вида пустой шкалы) и заполнения. Чем больше значение элемента, тем больше размер компонента-заполнения, прикрепленного к левому краю фона. Для минимального значения ширина компонента-заполнения устанавливается равной нулю. Для максимального значения — равной ширине фона.

– Slider: шкала с перемещаемой вдоль нее головкой. Положение головки устанавливает текущее значение. Головка может находиться только в фиксированных позициях, каждая из которых в порядке слева направо соответствует очередному значению соответствующего поля базового элемента Counter. Состоит из трех компонентов: статичного фона полосы, головки, которую пользователь может перемещать между левым и правым краями полосы, и заполнения полосы слева от головки. Левый край заполняющего компонента совпадает с левым краем полосы, правый край — с серединой головки.

– Stepper: элемент, позволяющий изменять текущее значение в заданном диапазоне с заданным шагом. Состоит из двух компонентов: нажатие на один их них увеличивает текущее значение на величину шага, нажатие на второй — уменьшает его на ту же величину. В реализации по умолчанию эти компоненты — кнопки с пиктограммами "плюс" и "минус", соответственно. Компоненты располагаются рядом друг с другом

в линию либо по вертикали, либо по горизонтали.

– Value Picker: элемент, аналогичный Stepper и в дополнение к внешнему виду последнего отображающий текущее значение в текстовом поле, расположенном между двумя кнопками. При помощи шаблонов кнопки могут быть заменены на любые элементы с аналогичным поведением (по аналогии с элементом Stepper). Третий компонент также может быть заменен на любой элемент, значение одного из полей которого при помощи простой связки должно быть ассоциировано с текущим значением Value Picker.

Поля ввода представлены двумя видами элементов: Text View и Rich Text View. Графическое представление для Rich Text View образуется рамкой, обрамляющей Content Presenter, тогда как Text View состоит из Overlay Panel, содержащей наложенные друг на друга Content Presenter (для ввода строки текста) и Text Label (для отображения текста по умолчанию поверх пустого поля). Оба элемента присутствуют в используемой в настоящее время версии протокола, но, как и вышеописанная Slide Panel, отличаются отдельными деталями внутреннего устройства.

Протокол поддерживает два вида кнопок: Button и Toggle Button. Обычная кнопка постоянно отображает текст или картинку и одинаково реагирует на нажатие. Toggle Button, в свою очередь, может находиться в двух состояниях, между которыми переходит по нажатию.

Кроме того, особняком среди шаблонизированных управляющих элементов стоит UserControl. Это единственный элемент из этой группы без реализации по умолчанию. Переопределяя такой элемент, то есть описывая дерево с корнем в узле UserControl и произвольными остальными элементами, пользователь протокола создает свой элемент.

Для шаблонизированных управляющих элементов предусмотрен механизм привязки, позволяющий задать взаимозависимость двух атрибутов — доступного извне свойства данного элемента и свойства какой-либо из частей внутреннего представления. Например, номер отобража-

емой страницы в Multiview Panel зависит от текущего значения используемого для построения этой панели Stepper. Причем извне обратиться к текущему значению Stepper пользователь протокола не может. При изменении одного из пары связанных свойств второе изменяется автоматически.

Пользовательские управляющие элементы — это переопределенные шаблонизированные элементы. Переопределение может быть осуществлено как непосредственно в ядре сервера Ubiq Mobile (например, к этому типу относится группа переключателей (Checkbox), задаваемая как стековая панель, содержащая пары из поля, в которое можно установить флажок (реализовано как рамка с заданной ввиде геометрического пути пиктограммой выбранного варианта), и надписи), так и в пользовательском коде.

3.3. Связки

Простые связки задают зависимость между значением параметра объекта, в котором они определены, и значением другого параметра того же типа (другого объекта или того же самого). Такая связка описывается идентификаторами связываемых параметров, а также идентификатором элемента-источника, из которого берется второй параметр.

Связки с данными используются для установления соответствия между шаблоном однотипных элементов и источником данных. Источник данных в этом случае состоит из последовательности групп значений, i-е значение в каждой группе имеет один и тот же тип и ассоциируется с одним и тем же тегом. Описание связки включает идентификатор параметра в шаблоне и тег в источнике данных, указывающий, какое именно значение из каждой группы будет использоваться для данного параметра при наполнении шаблона.

Связка может быть односторонней, то есть забирать данные из источника, но в источник обратно не передавать, и двухсторонней. В последнем случае изменение параметра в объекте также приводит к изменению параметра в источнике. Набор связок может быть указан у

элемента любого типа при перечислении свойств. Контроль за совпадением типов у связываемых параметров возлагается на сервер.

3.4. Стили

Под стилем подразумевается некоторый набор атрибутов с заданными значениями, на который могут ссылаться графические управляющие элементы. Описание стиля передается в виде отдельной вершины дерева элементов, для которой указываются входящие в данный стиль атрибуты и соответствующие им значения. Кроме того, стиль может указывать на другой стиль, наследуя и, возможно, дополняя его значения. Ссылки на стили должны образовывать древовидную структуру. Проверка отсутствия циклических зависимостей возлагается на сервер.

Каждый управляющий элемент содержит свойство StyleID, указывающее на вершину, описывающую стиль, заданный для данного элемента.

При применении стиля учитываются следующие факторы:

- Если стиль содержит значение атрибута, отсутствующего у данного элемента, это значение игнорируется. Это позволяет не дублировать стили для элементов разных типов, которые должны быть выполнены в одинаковом оформлении.

- Если стиль содержит значение атрибута, уже заданного для данного элемента, то используется заданная до применения стиля величина.

4. Особенности реализации

В рамках данной работы реализация заключалась в поддержке расширенной версии протокола на эмуляторе Ubiq Mobile. Так как изменения протокола затрагивают только графическую часть, то общая структура эмулятора была оставлена без изменений.

4.1. Разрешение перекрестных ссылок

Основная сложность в реализации была представлена обработкой элементов, ссылающихся на другие узлы дерева управляющих элементов. Ранее таких элементов в протоколе не было, поэтому все дерево строилось за один рекурсивный обход. Теперь же источник данных может ссылаться на элемент в еще не созданном поддереве. Изменение порядка обхода поддеревьев не поможет решить проблему, так как в дереве могут присутствовать перекрестные ссылки. При наличии таких ссылок вне зависимости от порядка обхода возникнет ситуация, когда ссылка ведет в ту часть дерева, которая еще не преобразована из бинарного представления к древовидному.

В связи с этим было решено сохранить существующий обход, создающий структуру дерева, но не обрабатывать в процессе этого обхода новые элементы (источники данных и шаблоны), а просто сохранять их как свойства соответствующих элементов дерева. По завершении рекурсивного обхода (когда управление возвращается в обработку корневого элемента Screen), по новой рассматриваются узлы, содержащие связки и шаблоны. Теперь, когда построение дерева завершено, элемент по любой ссылке гарантированно находится в дереве либо в виде узла, либо внутри шаблона, то есть любая ссылка может быть корректно обработана. Перед обработкой каждого шаблона требуется также рекурсивно обойти поддерево, соответствующее значению поля Template, поскольку оно может содержать другие нераскрытые шаблоны.

4.2. Обработка шаблонов

Использование шаблонов требует перестроения дерева после рекурсивного обхода, поскольку изначально шаблон является свойством элемента, а в конечном итоге должен занять в дереве место реализации этого элемента по умолчанию. В то же время, если шаблон включает в себя элемент типа Items Presenter, то структурирование дочерних элементов Items Presenter определяется типом переопределяемого элемента. Таким образом, при обработке шаблона последовательно выполняются следующие действия:

- осуществляется поиск элемента типа Items Presenter в поддереве шаблона;
- в случае, если такой элемент не найден, то управление передается на последний шаг алгоритма;
- иначе:
  - элемент типа Items Presenter в поддереве шаблона заменяется на элемент, шаблон которого обрабатывается;
  - число дочерних элементов определяется по размеру заданного для элемента типа Items Presenter источника данных;
  - шаблон элемента типа Items Presenter копируется требуемое число раз (в соответствии с количеством дочерних элементов) при помощи виртуального конструктора;
  - поля созданных на предыдущем шаге копий наполняются из источника данных;
  - эти копии подвешиваются к вершине, заменившей элемент типа Items Presenter в дереве;
- поддерево текущего элемента заменяется в основном дереве на поддерево шаблона.

4.3. Реализация табличной панели

Обработка элемента Grid Panel делится на три стадии:

- преобразование панели и ее поддерева из бинарного представления в древовидное;
- вычисление размеров столбцов и строк панели;
- вычисление координат дочерних элементов панели и их расположение внутри таблицы.

В соответствии с протоколом размеры строк и столбцов могут задаваться либо числом с плавающей точкой, либо специальным значением Fill Parent (в качестве него используется величина Double.PositiveInfinity). Кроме того, по результатам разбора пришедшей с сервера команды известен общий размер панели. На основании общего размера Grid Panel, а также фиксированных ширин столбцов и высот строк вычисляется остаток по горизонтали и по вертикали. Далее остаток по горизонтали поровну распределяется между столбцами, ширина которых задана значением Fill Parent, а остаток по вертикали — между строками с высотой, заданной этим же значением.

Каждый дочерний элемент панели содержит четыре дополнительных параметра, определяющих его положение в таблице. В число этих параметров входят координаты левой верхней ячейки, занимаемой дочерним элементом прямоугольной подтаблицы (поля Grid Row и Grid Column), и размеры этой подтаблицы (поля Grid Row Span и Grid Column Span). Суммируя ширины входящих в эту подтаблицу столбцов и высоты входящих в нее строк, можно получить размер прямоугольной области, доступной для размещения дочернего элемента. Затем от этой области с каждой из сторон отрезаются полосы, соответствующие внешним отступам дочернего элемента (поле Margin). В зависимости от используемых способов выравнивания по вертикали и по горизонтали дочерний элемент, размер которого известен и не зависит от содержащей его панели, располагается с краю или по центру оставшегося пространства.

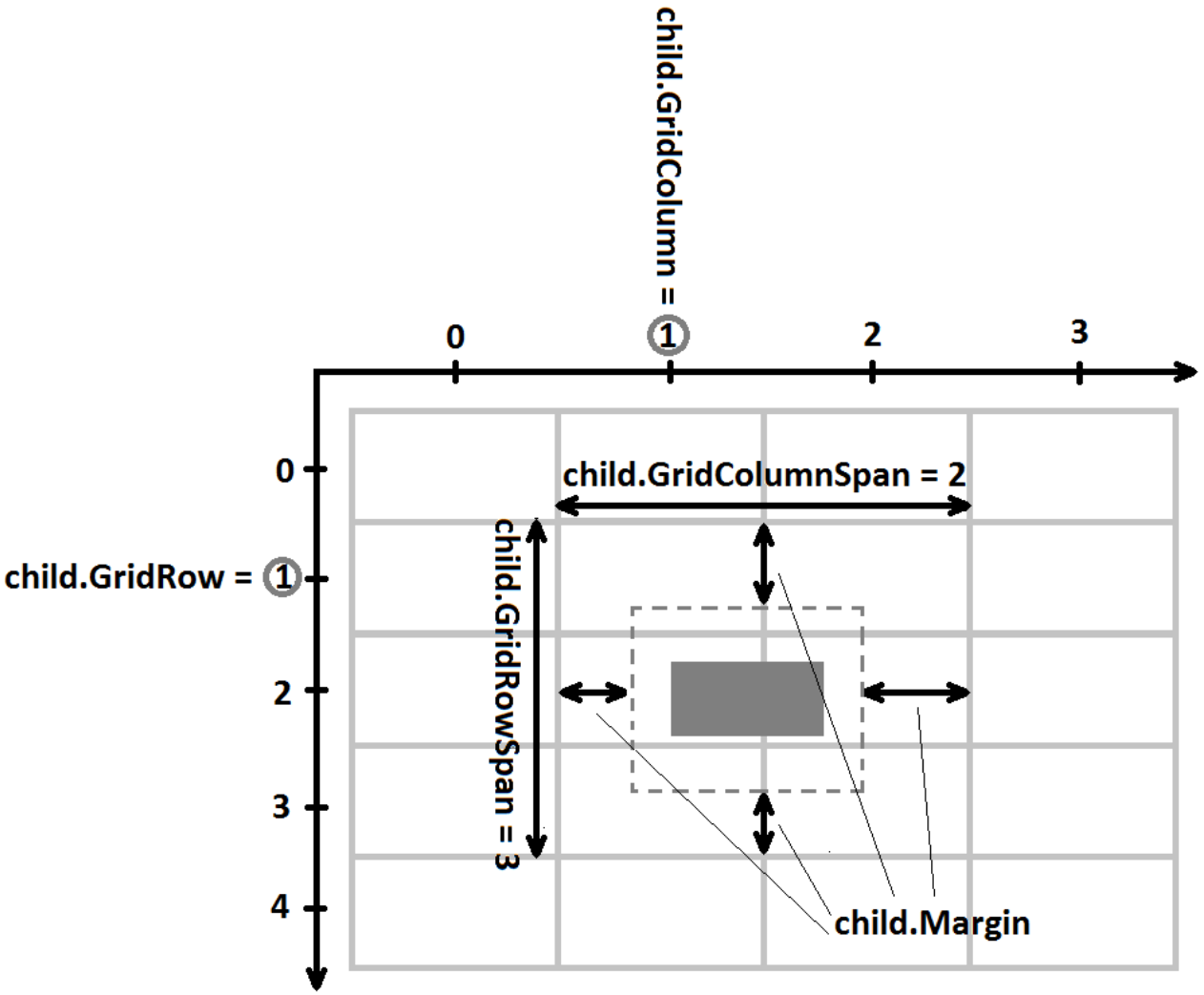

Рис. 1: Пример размещения дочернего элемента табличной панели

На рис. 1 показан пример размещения прямоугольника в подтаблице, занимающей три строки и два столбца. В данном случае значение полей Vertical Alignment и Horizontal Alignment равно Center, т.е. прямоугольник размещается в центре оставшегося после отрезания полос-отступов пространства, обозначенного прямоугольным пунктирным контуром. Координаты левого верхнего угла дочернего элемента вычисляются как сумма координат левого верхнего угла ячейки (Grid Row, Grid Column), размера соответствующего отступа (отступа слева для координаты по горизонтали и отступа сверху для вертикальной координаты) и половины свободной части ограниченной пунктирным контуром площади.

4.4. Реализация элемента Slider

Элемент Slider состоит из статичного компонента (полосы фона) и двух расположенных поверх него динамически изменяемых компонентов (полосы заполнения и ползунка). Slider может находиться в трех состояниях: неактивный, активный в фокусе, активный вне фокуса. По умолчанию элемент находится в неактивном состоянии.

При нажатии на ползунок (событие OnPreviewMouseLeftButtonDown) он переходит в состояние "активный в фокусе". При перемещении курсора в пределах полосы фона (событие OnPreviewMouseMove) ползунок смещается по горизонтали на расстояние, равное изменению горизонтальной координаты курсора (при этом вертикальное смещение курсора внутри полосы фона никак не учитывается). Одновременно с этим компонент, отвечающий за полосу заполнения, растягивается или сжимается по горизонтали так, чтобы его правый край проходил через центр ползунка. Левый край при этом остается прикреплен к левому краю полосы фона.

При перемещении курсора за пределы полосы фона (событие OnMouse Leave) элемент переходит в состояние "активный вне фокуса". В том случае, если курсор вернется на полосу фона, то Slider снова станет "активным в фокусе", а потому ползунок переместится по горизонтали к новой позиции курсора, а размер компонента, отвечающего за полосу заполнения, соответствующим образом изменится. При отпускании левой кнопки мыши (событие OnPreviewMouseLeftButtonUp) вне зависимости от того, в фокусе элемент или нет, Slider возвращается в неактивное состояние.

5. Описание процесса тестирования

Для тестирования изменений в эмуляторе использовалось специальное приложение, демонстрирующее новые возможности протокола. Это приложение состоит из основного экрана, отображающего список из элементов "Test 1", "Test 2", и т.д., первые два из которых активны, и дополнительных экранов, открывающихся по нажатию на соответствующий активный элемент списка.

Список на основном экране использует механизм шаблонов для переопределения внешнего вида. В данном случае шаблон используется для указания того, что список (элемент List View) заменяется на ячейку (элемент Cell), которая, в свою очередь, содержит элемент типа Items Presenter. Источник данных элемента Items Presenter содержит восемь строк (со значениями "Test 1", "Test 2", и т.д.), а шаблон — неизменяемое текстовое поле (элемент Text Label). При помощи простой связки каждая строка в источнике данных привязывается к полю Text соответствующего дочернего элемента типа Text Label.

Первый тест использует табличную панель (элемент Grid Panel). Дочерние элементы таблицы занимают как одиночные ячейки, так и содержащие несколько ячеек прямоугольные области. Для наполнения ячеек используются геометрические примитивы — прямоугольники.

Второй тест содержит ползунок (элемент Slider), использующий не стандартную, а переопределенную через шаблон реализацию. Для каждого из дочерних элементов шаблона указана его роль (ползунок, полоса для перемещения ползунка, заполнение полосы слева от ползунка).

Заключение

В рамках данной работы были получены следующие результаты:

- Разработана расширенная версия протокола с поддержкой гибкой настройки существующих элементов, создания пользовательских управляющих элементов, привязки источника данных к интерфейсу и стилей.
- Реализована поддержка расширенной версии протокола для эмулятора Ubiq Mobile.
- Работа эмулятора проверена на тестовом приложении.

Список литературы

[1] К.Н. Невоструев. Разработка и реализация терминального протокола и клиента в системе Ubiq Mobile для платформы iOS // Дипломная работа, Санкт-Петербургский государственный университет. — 2012.

[2] Xamarin framework documentation. — 2017. — URL: `https://developer.xamarin.com/guides` (online; accessed: 16.04.2017).

[3] PhoneGap framework documentation. — 2017. — URL: `http://docs.phonegap.com/` (online; accessed: 16.04.2017).

[4] Qt library documentation. — 2017. — URL: `http://doc.qt.io/` (online; accessed: 16.04.2017).

[5] Nathan Adam. WPF 4 Unleashed. — Sams Publishing, 2010. — 825 p. — ISBN: 0672331195.