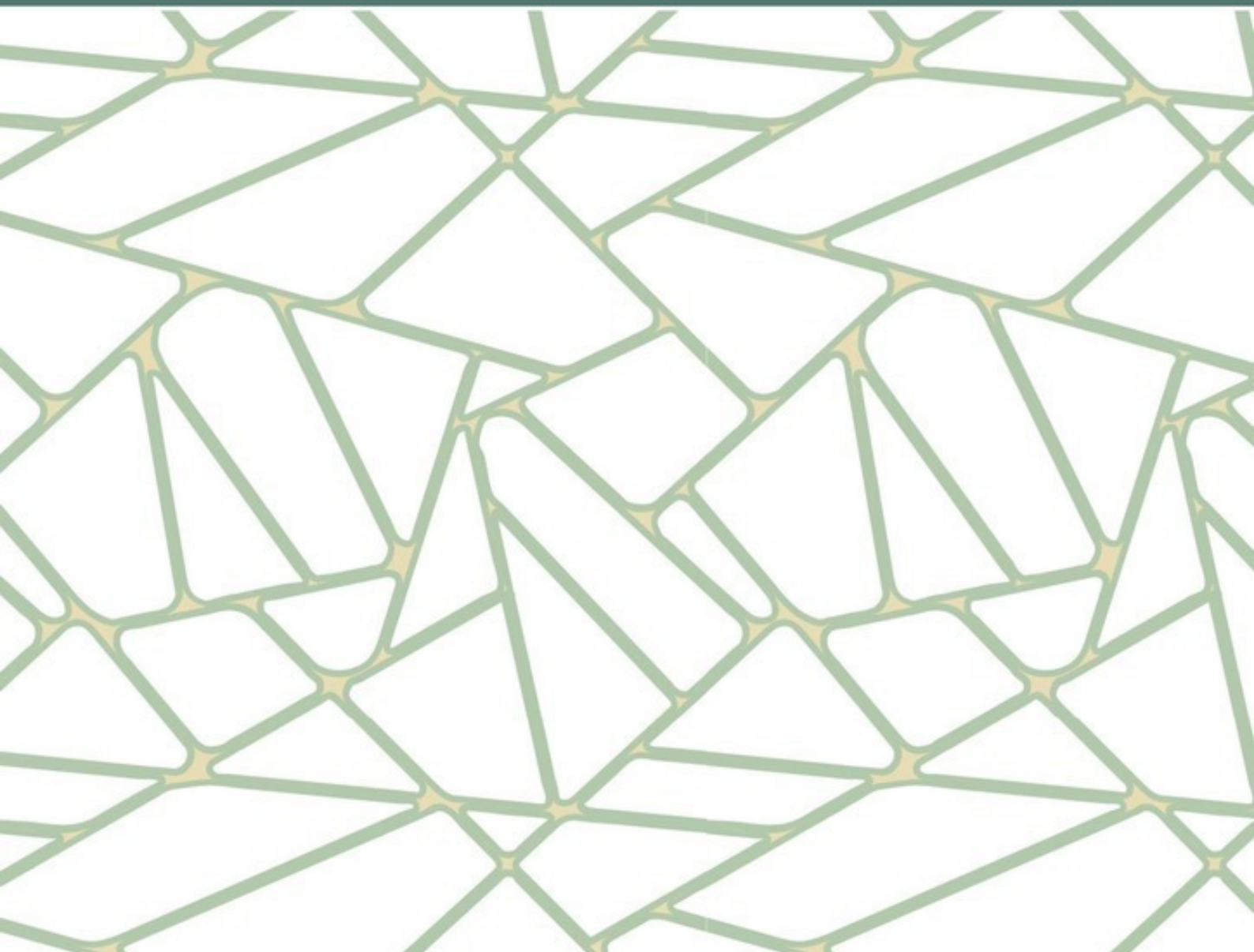
Виталий Иванович Донцов

Delphi: реальности программирования для смартфонов



Виталий Донцов

Delphi: реальности программирования для смартфонов

«Издательские решения»

Донцов В. И.

Delphi: реальности программирования для смартфонов /
В. И. Донцов — «Издательские решения»,

ISBN 978-5-00-605271-0

Книга посвящена практике работы мобильных приложений в среде Delphi 10.3. До настоящего времени нет практических руководств в этой области, при этом Delphi 10.3 значительно отличается от более ранних версий, Описывается последовательно создание приложений и рабочие компоненты, которые реально работают в данной среде, особенности и оптимальные режимы использования, описание основных свойств и функций, реальные примеры, что позволяет использовать книгу и новичкам.

ISBN 978-5-00-605271-0

Содержание

Delphi: реальности программирования для смартфонов

Виталий Иванович Донцов

ISBN 978-5-0060-5271-0
Создано в интеллектуальной издательской системе Ridero

Введение

Язык программирования Delphi представляет собой объектно-ориентированный высокоуровневый язык программирования, направленный на написание прикладного программного обеспечения и является диалектом языка Pascal. Возможность легко и просто создавать приложения для персональных компьютеров на платформе Windows с прекрасным дизайном (типа Microsoft Office) снискала любовь к нему многих.

Вышедшая в 2011 году версия Delphi XE2 добавила компилятор Win 64 и поддержку операционных систем фирмы Apple (MacOS X, iOS), а в 2013 году Delphi XE5 позволила создавать приложения для устройств на платформе Android. С переходом на XE6 было исправлено более 500 багов, однако, все имеющиеся версии все еще очень далеки от совершенства. В настоящее время наиболее приемлемой является версия Delphi 10.3, в различном исполнении: «Сиэтл», «Берлин», «Пекин»… В настоящей книге описан опыт работы с вариантом «Сиэтл», позволяющий, по замыслу авторов, создавать кросплатформенные приложения с одним кодом для разных платформ.

Работа по созданию кросплатформенных приложений, однако, имеет много особенностей, а каждая версия, по существу, заново переписывает многие свойства и функции, так что руководства для XE5, 6, 10 очень часто дают неверную информацию и приводимые примеры зачастую не работают. К тому же, для тех, кто работал на Delphi 6 и 7, новостью будут как новый язык, названные и библиотека которого – FireMonkey, так и новые подходы и идеи программирования.

Все это затрудняет работу с Delphi 10.3, которая является, видимо, наиболее приемлемой из современных вариантов Delphi для создания Android приложений, но все еще имеющей множество багов и особенностей работы, которые как раз и будут описаны в данной книге, целью которой является описание наиболее важных и реально работающих компонентов для создания прикладных программ для современных смартфонов.

1. Delphi 7 или Delphi 10.3?

Для тех, кто ориентирован на создание красивых, мощных и быстродействующих программ для персональных компьютеров платформы Windows, выбором, видимо, является версия Delphi 7, к которой вернулись большинство программистов из более поздних версий. Она является, по мнению большинства, наиболее стабильно работающей и наиболее дружественной программой, на которой можно создавать практически любые прикладные программы для персональных компьютеров.

Интересно, что при создании кросплатформенных приложений одновременно с вариантом для Android создается и вариант для Windows, причем значок приложения позаимствован у Delphi 7.

Различие программ, созданных на Delphi 7 и 10.3 определяется различием платформ и, главным образом, экрана. Экран персональных компьютеров не только больше, но и ориентирован как «Ландшафтный», тогда как «Портретный» экран смартфона, вытянутый практически в 2 раза к ширине, не дает возможности каких-либо украшений и даже надписи приходится делать мелкими и едва читаемыми. Также в Android варианте недоступны многие функции, привычные для Delphi 7, например, популярное «Меню» отсутствует, как и текстовый редактор «Rich Edit», таблицы совершенно изменены, а графики урезаны до минимума. Таким образом, лучше использовать дружественный отшлифованный интерфейс Delphi 7 для ПК, тогда как только необходимость создания приложения для смартфона делает нужным использование Delphi 10.3, причем создаваемая параллельно версия для Win 64 (или 32) совершенно примитивна и практически не нужна.

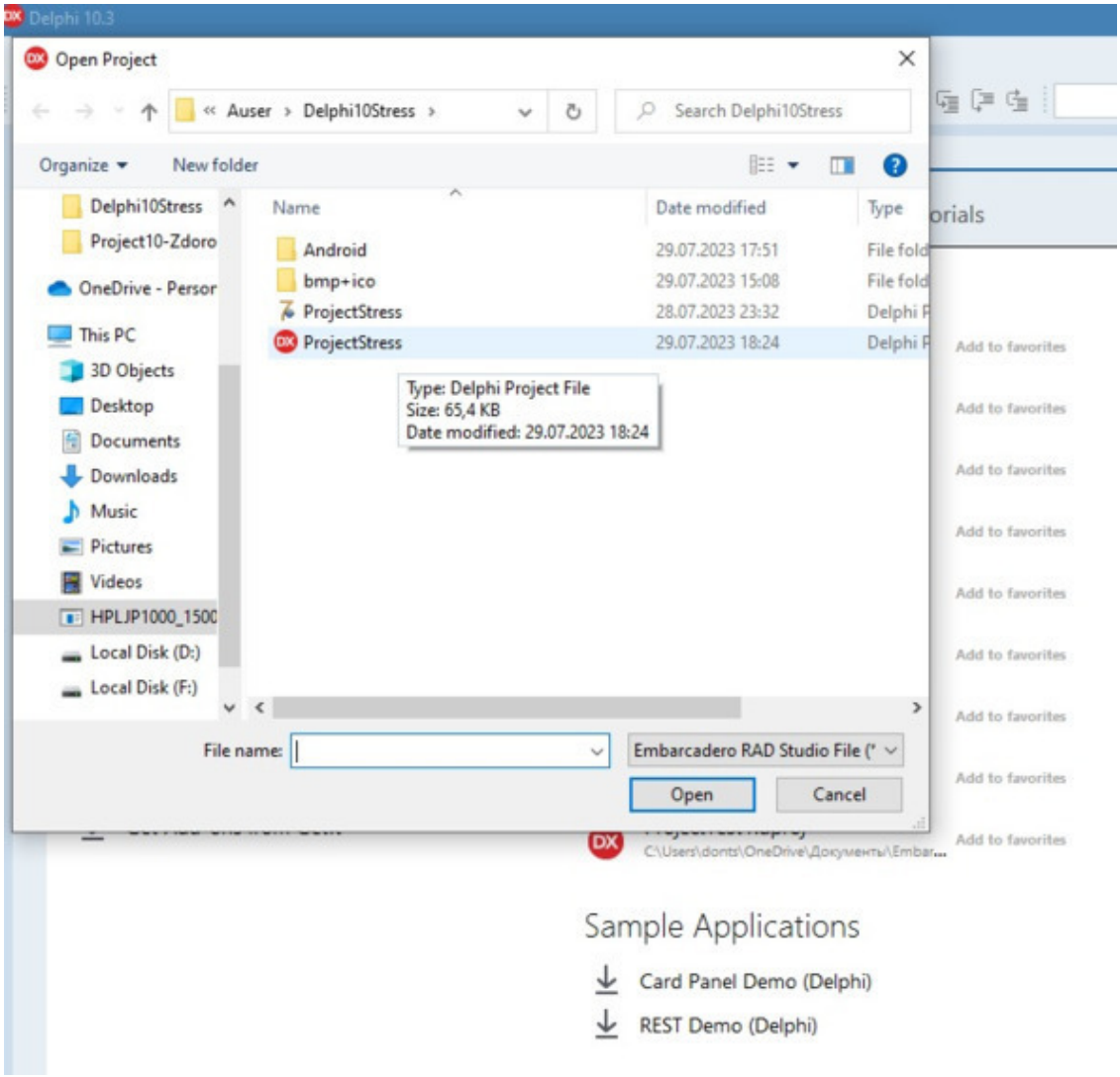


Рис. 1. Программы Delphi 10.3 создаются сразу в вариантах Android и Win 64.

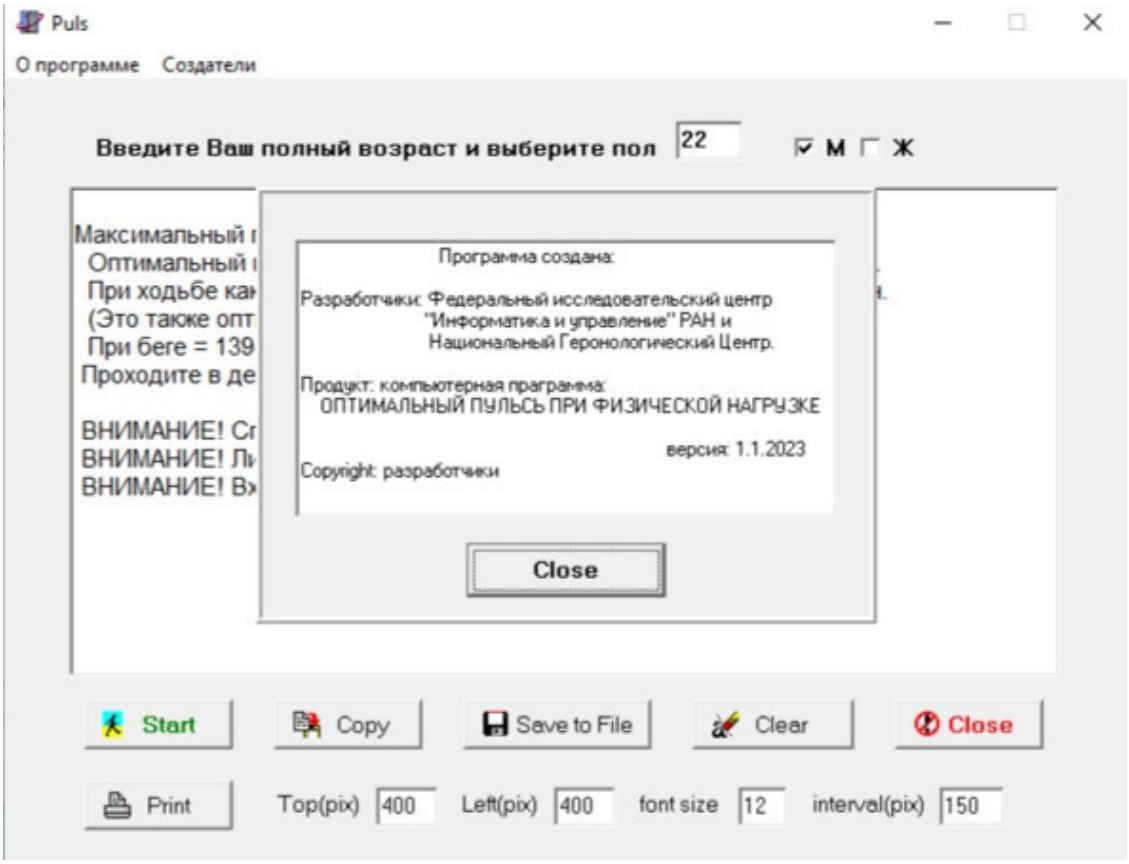


Рис. 2. Прикладная программа «Оптимальный пульс» для ПК на Windows.

Так видим, для ПК имеется и меню, и больше возможностей обработки текста, сохранения в файл, распечатки, большой экран и большой текст, а также обычные значки на кнопках, для которых на смартфоне нет места.

Надо заметить, что **копирование** текста с возможностью вставить в текстовый редактор (или в e-mail себе для передачи в свой ПК) имеет место в обоих приложениях, а **скрин** экрана доступен в Android как функция платформы и может затем использоваться и передаваться как обычное фото.

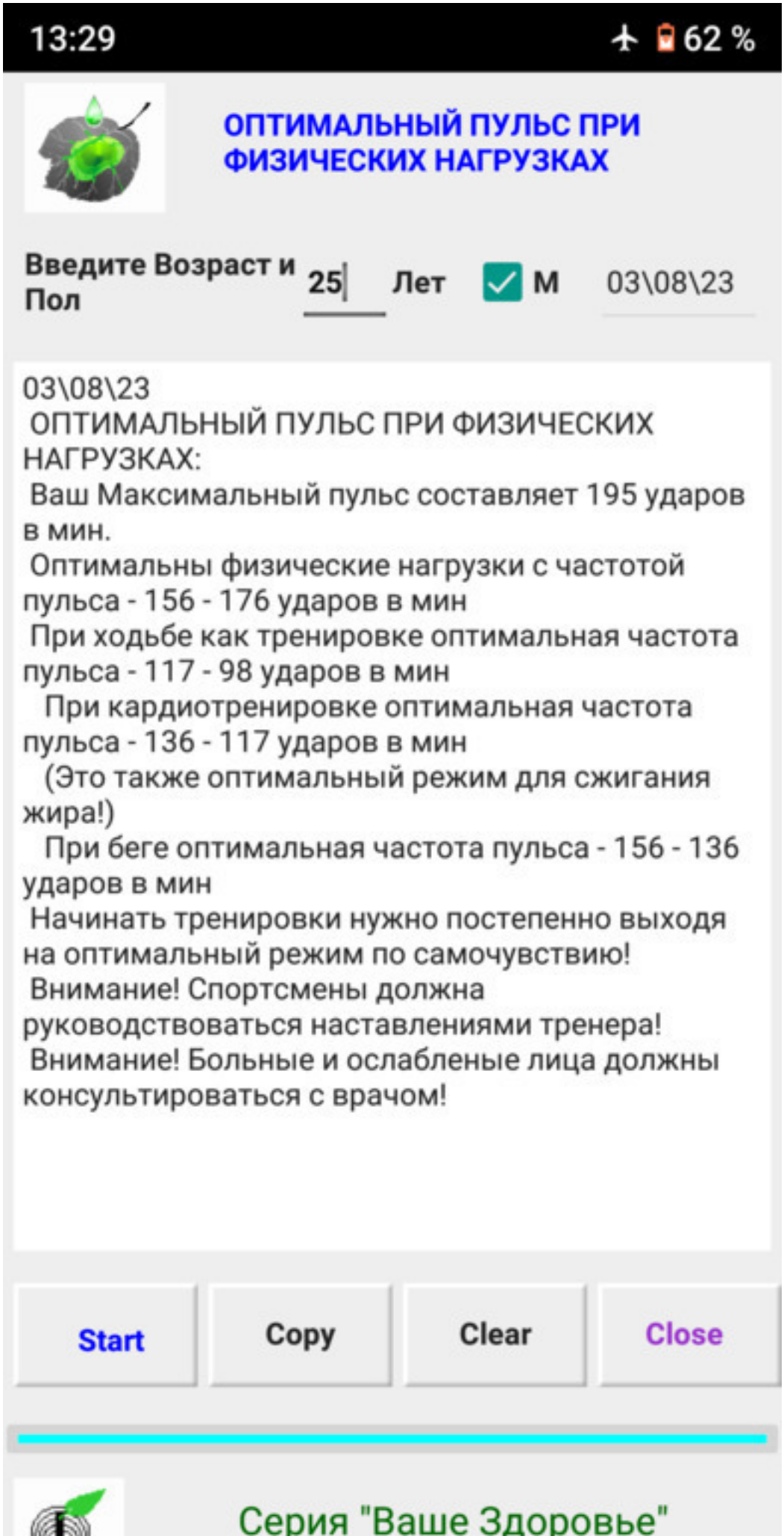

13:29
62 %
ОПТИМАЛЬНЫЙ ПУЛЬС ПРИ ФИЗИЧЕСКИХ НАГРУЗКАХ
Введите Возраст и Пол
25
Лет
M
03\08\23
03\08\23
ОПТИМАЛЬНЫЙ ПУЛЬС ПРИ ФИЗИЧЕСКИХ НАГРУЗКАХ:
Ваш Максимальный пульс составляет 195 ударов в мин.
Оптимальны физические нагрузки с частотой пульса - 156 - 176 ударов в мин
При ходьбе как тренировке оптимальная частота пульса - 117 - 98 ударов в мин
При кардиотренировке оптимальная частота пульса - 136 - 117 ударов в мин
(Это также оптимальный режим для сжигания жира!)
При беге оптимальная частота пульса - 156 - 136 ударов в мин
Начинать тренировки нужно постепенно выходя на оптимальный режим по самочувствию!
Внимание! Спортсмены должна руководствоваться наставлениями тренера!
Внимание! Больные и ослабленые лица должны консультироваться с врачом!
Start
Copy
Clear
Close
Серия "Ваше Здоровье"

Рис. 3. Прикладная программа «Оптимальный пульс» на Android для смартфона.

2. К началу мультиплатформенного программирования

2.1. Особенности Delphi 10.3

Наиболее отличающимся от обычного Delphi, видимо, является язык и библиотека «FireMonkey», которую авторы поднимают до уровня платформы. Задумка новой платформы была в возможности кросс-платформенного кода, одинакового для разных платформ, от Windows до Android, и значительное расширение возможностей: введение дополнительных наборов классов компонентов и сервисных интерфейсов, написанных на языке Delphi, в том числе для 3D-приложений и высококачественной графика. На деле, однако, при реальной разработке приложений для Android сразу приходится сталкиваться с отсутствием некоторых привычных и важных компонентов (меню, RichEdit, урезанных таблиц, отсутствием ряда свойств, например, цвета у Панели, неприглядного изначального вида Кнопок, едва видимого Edit и многого другого).

К тому же, нестабильность работы приводит к тому, что пропадают ряд свойств компонентов, вдруг перестают работать элементарные функции, время работы Timer растягивается в 2 раза, на сохранение файлов накладываются ограничения и запреты самого смартфона, в зависимости от экрана смартфона компоненты приложения могут вылезать за пределы экрана, а функция скроллинга не включаться, что не дает возможности в Memo прочесть текст до конца, клавиатура накладывается на компоненты и не видно, что записывается, и пр., включая крайне медленную работу приложений и их большой объем. Тем не менее, возможность создавать работающие приложения для Android имеется, как и возможность выкладывать их в мировую сеть через Google Play Market, а для привычных в программировании к Delphi это оптимальный вариант создания APK программ для смартфона, учитывая, что это прикладные, а не игровые программы, для которых он не разрабатывался.

2.2. Начало работы и создание Проекта (приложения) в Delphi 10.3

Для начала работы следует скачать, найдя в Яндекс или Google, установочную программу Delphi 10.3 в любом варианте. Наиболее доступно, видимо, «Сиэтл». Запустить установку – это все, что требуется, в отличие от других программ, все происходит само, с созданием значка на экране и открытием самой программы.

При открытии мы видим весьма непрезентабельную заставку наполовину из рекламы, которую спокойно можно опустить и сразу открыть файл создания приложения: *файл – новый – мультиплатформенное приложение* – один из полуготовых вариантов (обычно открывают «*Blank Application*»: пустую заготовку).

Сразу создается, как обычно, форма (*Form1*), на которой и будут располагаться различные компоненты работающей программы.

Здесь же видны основные части программы для работы с компонентами.

Слева: *Structure* (Структура взаимоподчинения компонентов, пока что представлена одной Form1), *Object inspector* (Инспектор объектов для выбора свойств объектов).

Справа: Pellete (Палитра сгруппированных компонентов) и отображение Project Group с наиболее важными функциями: *Build Configeration* (*Debug* и *Realese*) – для окончательной конфигурации и выпуска программы.

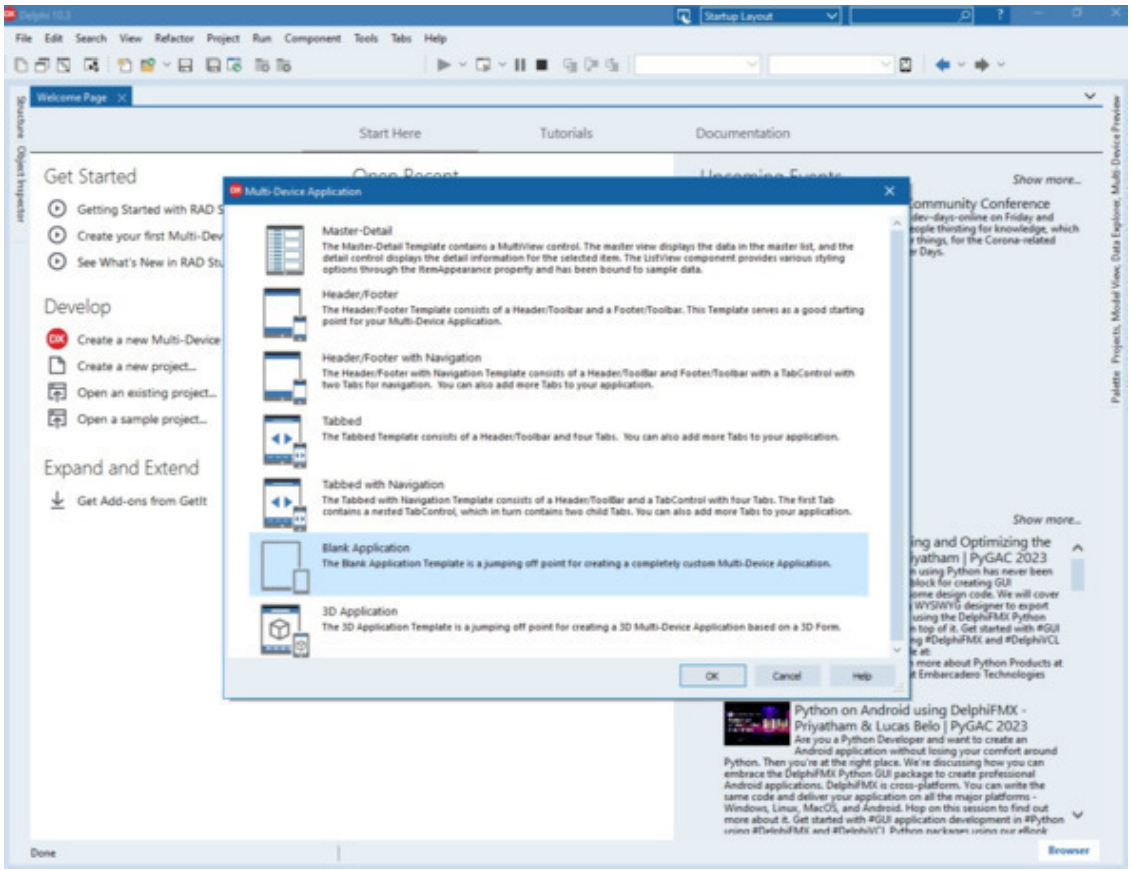


Рис. 4. Создание мультиплатформенного приложения.

Вверху: ряд кнопок для управления, наиболее важны: *File* – для создания, сохранения и открытия приложения; *Project* – с функцией *Option* для настройки приложения (значок, вид и ряд др.) и зеленые кнопки запуска работы программы (с предварительным debug и без него). Запуск программы для промежуточных уточнений и исправления ошибок (которые всегда имеются) реально проводить можно сразу же на подсоединенном к ПК смартфоне (в режиме «разработчика»), поэтому первым делом такой режим нужно настроить, предварительно выбрав в 2-х верхних окошках режим «Android» и «Master» (для кросплатформенного приложения).

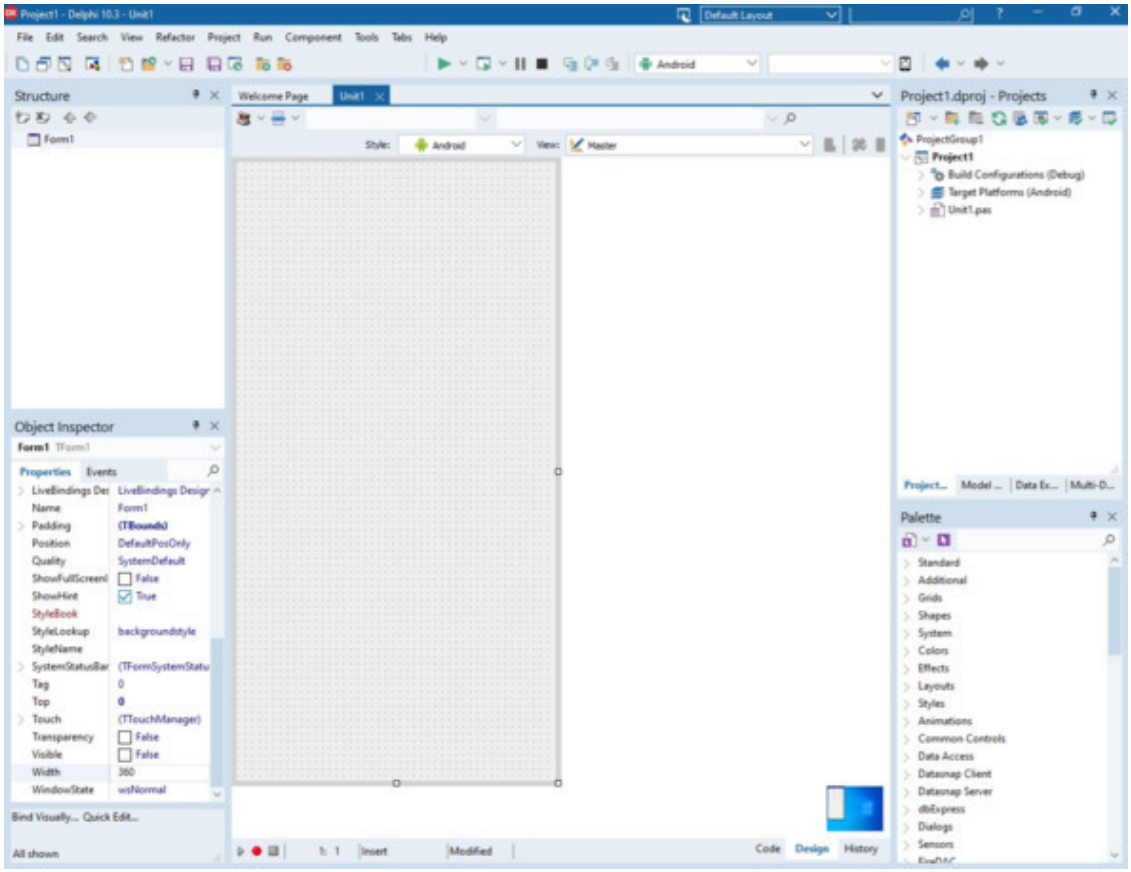
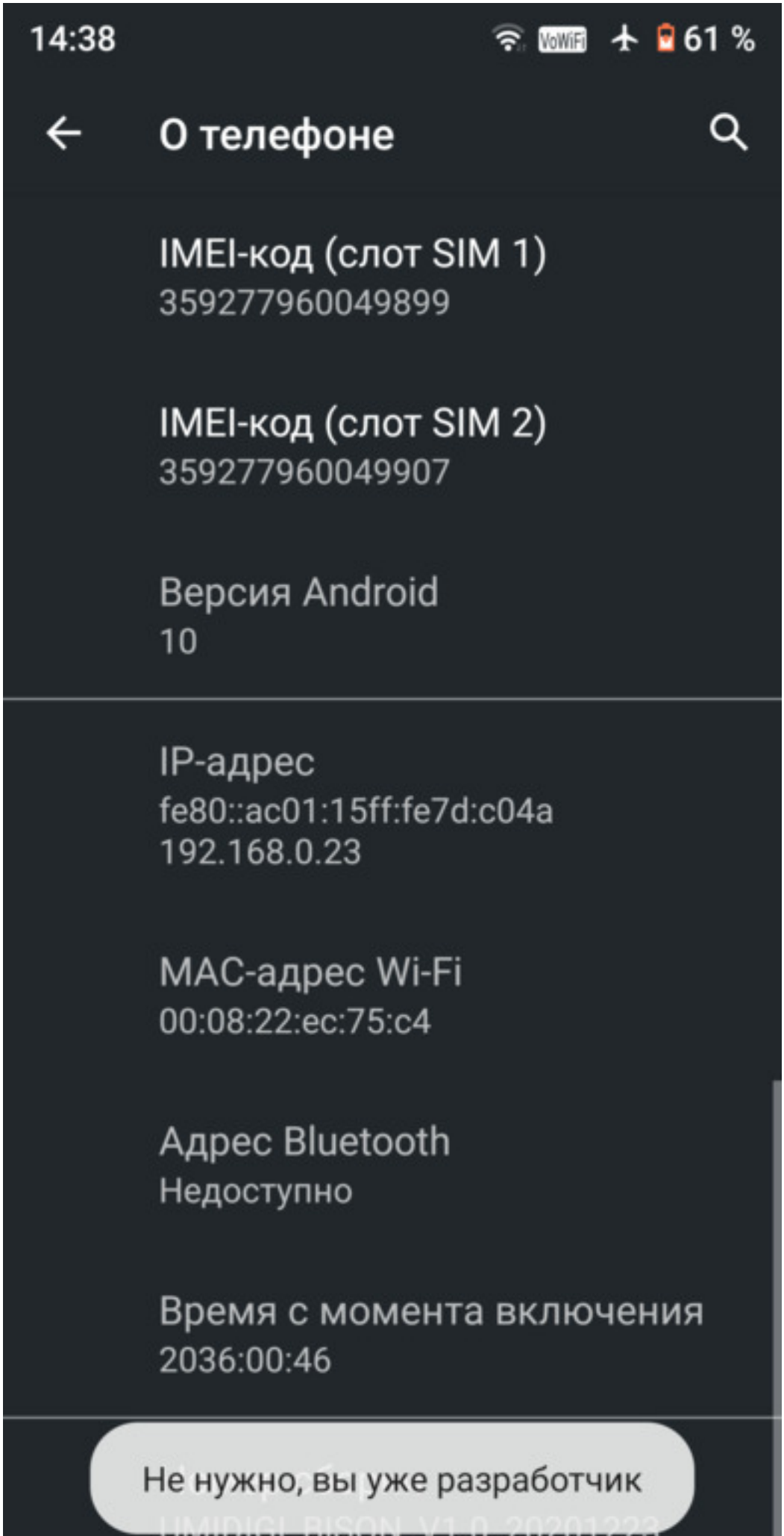


Рис.5. Открываемая первичная форма для расположения компонентов приложения.

2.3. Настройка смартфона для работы с создаваемым приложением

Для входа в режим разработчика нужно войти в «*Настройки*» смартфона и в разделе «*О телефоне*» нажать несколько раз на последний пункт «*Номер сборки*» для появления сообщения «Вы стали разработчиком», затем перейти в раздел «*Система*» и «*Дополнительно*», где появился пункт «*Для разработчиков*», нажав на который пройти почти до конца до пункта «*Отладка*» и выбрать флажок «*Отладка по USB*». Теперь, подключив смартфон к компьютеру, мы увидим вверху в окошке название смартфона.


14:38
VoWiFi
61 %
О телефоне
IMEI-код (слот SIM 1)
359277960049899
IMEI-код (слот SIM 2)
359277960049907
Версия Android
10
IP-адрес
fe80::ac01:15ff:fe7d:c04a
192.168.0.23
MAC-адрес Wi-Fi
00:08:22:ec:75:c4
Адрес Bluetooth
Недоступно
Время с момента включения
2036:00:46
Не нужно, вы уже разработчик

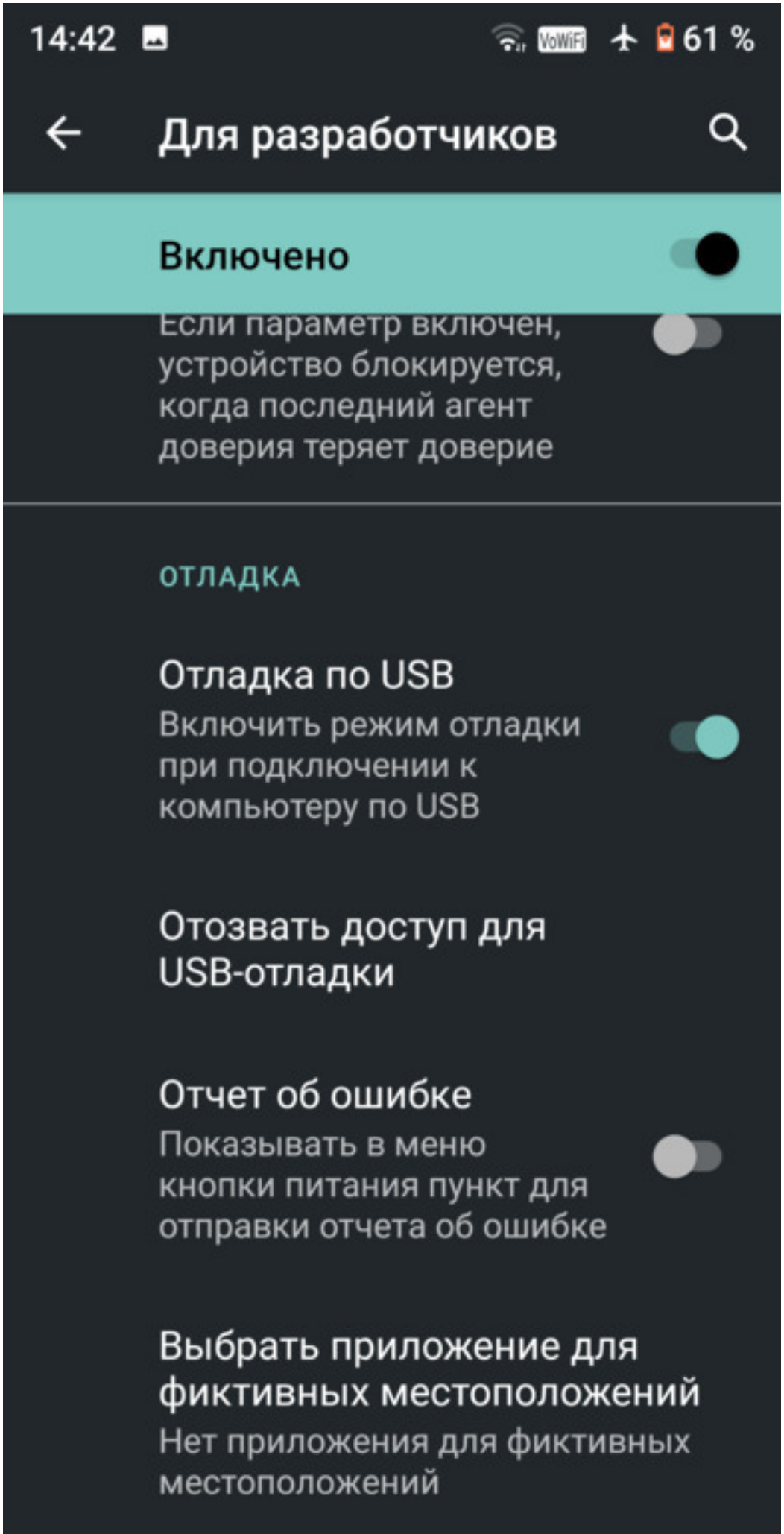

14:42
61 %
Для разработчиков
Включено
Если параметр включен, устройство блокируется, когда последний агент доверия теряет доверие
ОТЛАДКА
Отладка по USB
Включить режим отладки при подключении к компьютеру по USB
Отозвать доступ для USB-отладки
Отчет об ошибке
Показывать в меню кнопки питания пункт для отправки отчета об ошибке
Выбрать приложение для фиктивных местоположений
Нет приложения для фиктивных местоположений

Рис. 6 и 7. Включение режима «разработчик» на смартфоне.

2.4. Основная форма приложения (Form1)

При начале работы с созданием приложения, сразу нужно определиться с несколькими моментами. Во-первых, сразу изменить размеры и форму **Form1**, в соответствии с формой смартфона. Так как размеры смартфона практически соотносятся 1:2 можно выставить размеры, например: 360 * 700. Все остальные настройки можно сохранить неизменными.

При желании можно изменить цвет формы (свойство *Fill – Color*; а также получить градиентную заливку: *Brush – Gradient*, свойство gradient должно быть и в *Kind* и в *Gradient*).

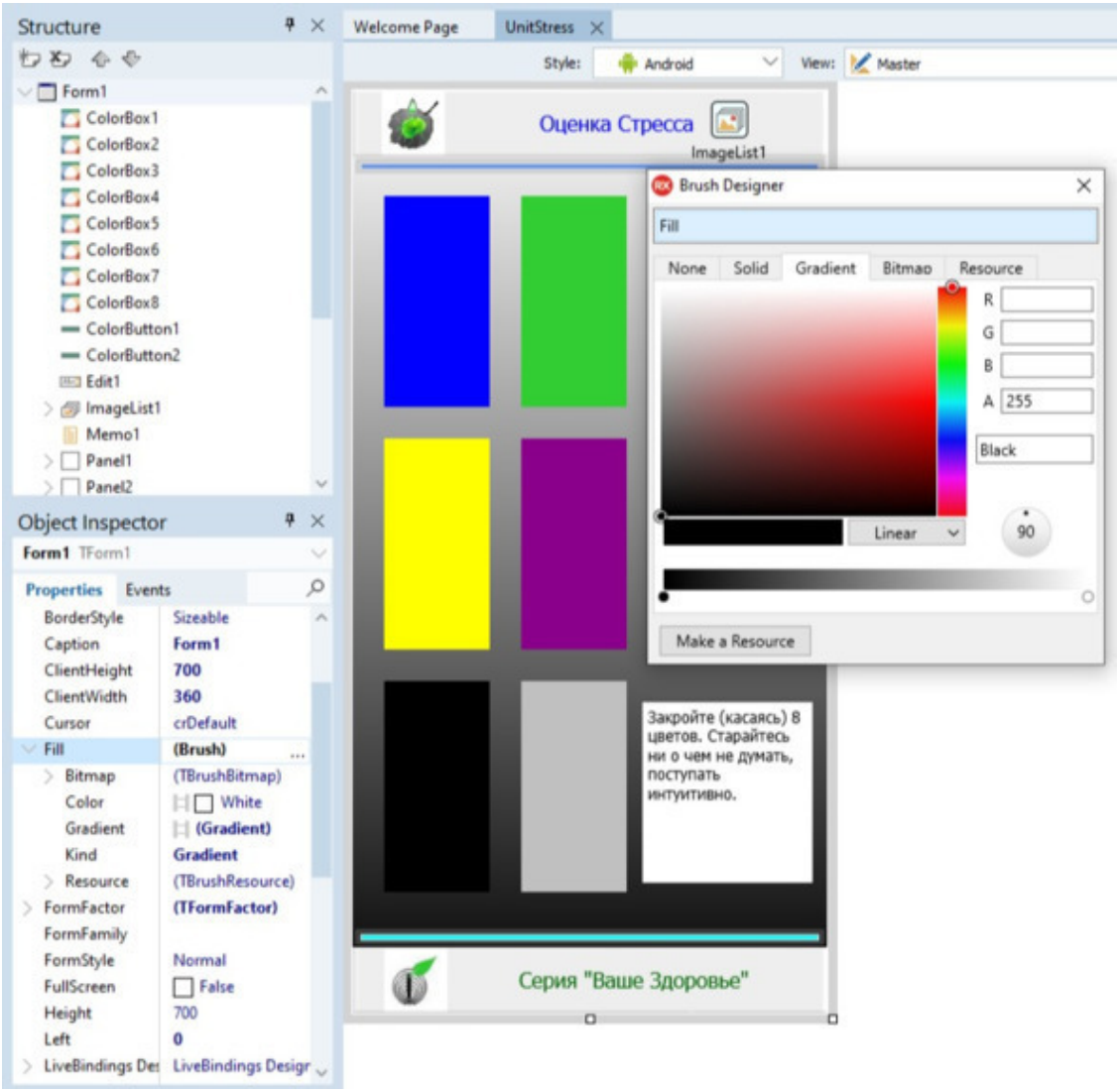


Рис.9. Заполнение цветом и градиент цвета для компонентов.

На специальной панели Brush Desiner с помощью 2-х ползунков можно выставить любой цвет и градиент, а также повернуть градиент на $+90^0$.

Нужно также поставить галочку для *Full Screen* и для *Position* выставить *Screen Center*. Другие все «красивости» для прикладных программ излишни.

После создания Формы на нее перетаскиваются из Палитры все необходимые компоненты: Кнопки (*Button*), Метки (*Label*), Окошки (*Edit*), компоненты для отображения текста (*Memo*, при том, что Rich Edit не доступен для Android) и Базы данных (Таблица: в Delphi 10.3 два типа таблиц – обычная *StringGrid* для строчных данных и специальная *Grid* для любых

типов данных). В основном используются компоненты из групп *Standard* и *Grids*. Широко используются компоненты *Panel* и *Layout* как контейнеры, обычно для выравнивания других компонентов, которые на них располагаются.

2.5. Сохранение и запуск работы приложения

На первом этапе создания приложения следует его **сохранить в выбранной папке** (сделайте папку с понятным названием), при этом все директории и папки не должны иметь русских букв. На каждом этапе создания приложения и перед его запуском следует его сохранять!

Для проверки программирования можно в верхнем меню *Project* выбрать *Build Project*, при наличии ошибок он выдаст анализ программы и выделит ошибки с указанием на них и описанием в нижней панели.

Подключив смартфон можно, нажав на зеленую кнопку *Run* увидеть как работает программа на настоящий момент. Обработка, передача и запуск программ обычно идет достаточно долго, что приводит к выдаче предупреждения о невозможности запуска в виду уже загруженности устройства, на это не нужно обращать внимание.

При запуске программы создается обычный для Android **файл APK**, который располагается в: *Android/Debug (Release) /Имя/bin/ИмяПроекта. APK.*

2.6. Подготовка приложения к выпуску

Для окончательной готовности приложения следует провести ряд действий:

– Перейти в *Project – Option – Version Info* и изменить название программы (по умолчанию равна названию проекта) – пункт *label*.

– На *Uses Permissions* можно выбрать перечень разрешений запрашиваемых от системы (изначально все пусты!), но лучше так все и оставить.

– Можно сменить Значок – пункт *icons* (создать и загрузить несколько – от 36x36 до 144x144).

– Также в разделе *Otientation* можно выбрать *Portrait*, или оставить так – если Вы готовы сделать два приложения для разной ориентации экрана.

Наконец, нужно создать **Подпись – Сертификат**: *Project – Options – Provisioning* – готовность к работе изъявит кнопка *New Keystore*, на щелчок открывается помощник по созданию хранилища ключей: название файла хранилища (*.*keystore*), *пароль* доступа к хранилищу, *псевдоним* ключа (ключевую пару открытого и секретного ключа), *пароль* к псевдониму и *период действия* ключа (по умолчанию 9132 суток). После завершения получим требуемый сертификат, предназначенный для подписи файла.

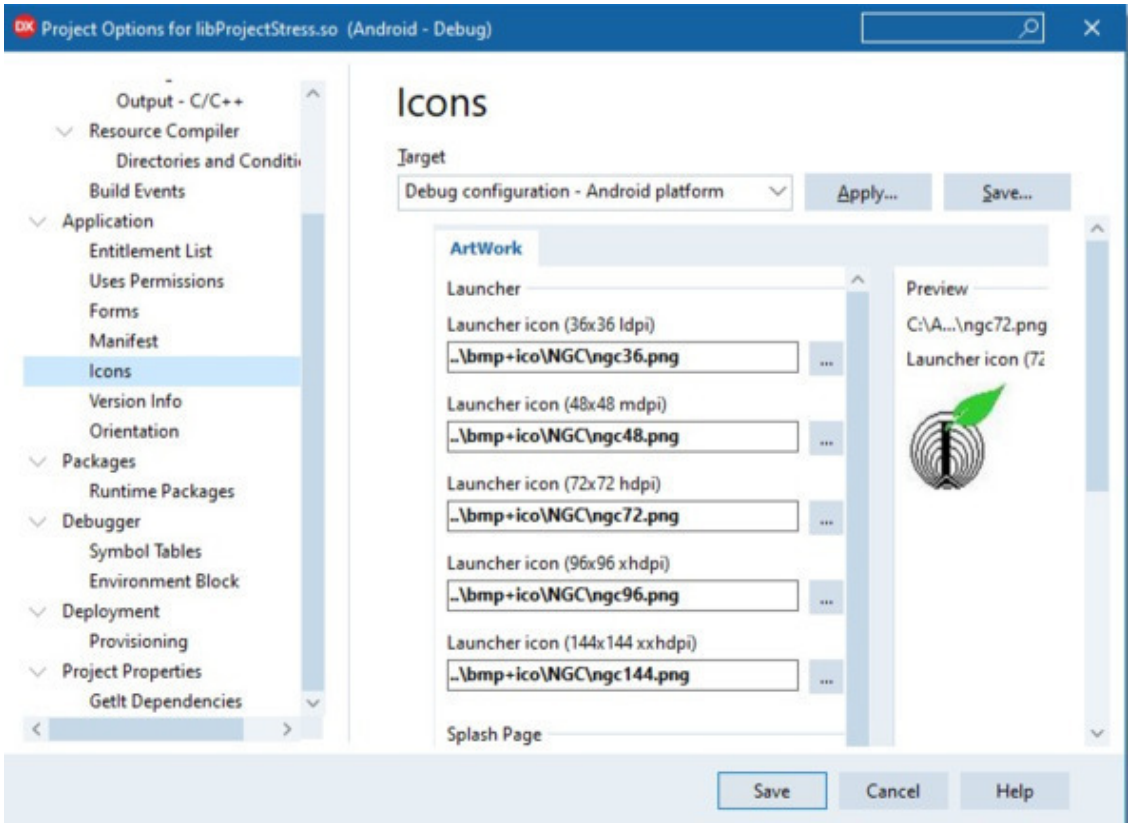


Рис. 10. Смена значка приложения.

Для выкладывания приложения в **Google Play** нужно найти страницу разработчиков Google; ввести аккаунт (например, который использовали при регистрации своего устройства на базе Android) и небольшую плату.

После прохождения платежа окажетесь в своём кабинете, позволяющем загружать на Google play приложения.

Завершив создание профиля разработчика, Вы получаете возможность отправить своё приложение на сервер Google, процесс начинается с щелчка по кнопке Upload Application.

После загрузки заполните профайл вашего приложения, даже поля, помеченные как необязательные (поможет Google продвигать приложение на рынке).

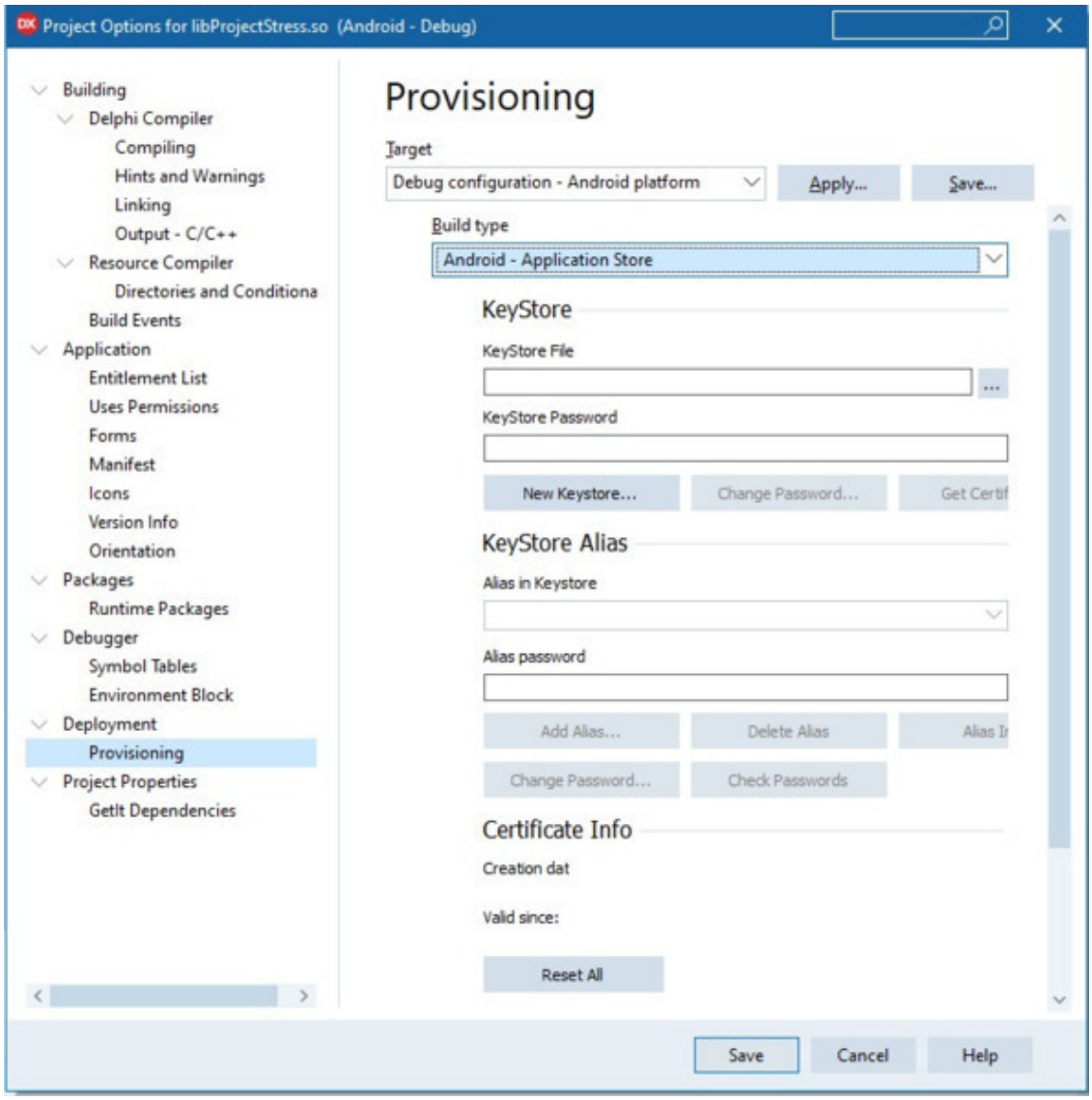


Рис. 11. Создание сертификата приложения.

2.7. Особенности свойств компонентов для Android

Главные особенности при программировании для Android связаны с различиями смартфонов – диагонали и разрешения экрана, что сбивает настройки компонентов и может вести к выходу их за пределы экрана.

Для разных компонентов в режиме *Master* используется *Scale* для компонентов чтобы не меняя размера шрифта изменить его для любого устройства.

Также широко используются компоненты палитры *LayOut*, особенно *GridPanelLayout* и *FlowLayout* – компоненты располагаются рядом с фиксированным расстоянием, изменение одного отодвигает другие; *GridLayout* – сетка, обычно используется для нескольких одинаковых компонентов. В редакторе *Structure* следует перетащить мышью компонент в подчинение *Layout*. Такой прием используется и для придания свойств компонентам с палитры *Effects*. Компонент *Panel* также используется как контейнер для других компонентов.

Также используется *Плавающая верстка* – привязывая один компонент к границам другого методом *Align,* при этом важна последовательность создания компонентов и их Align.

Внутри контейнера используется свойство *Margin* для фиксирования расстояния от границ контейнера, а также свойство *Position*.

В Delphi 10.3 широко используются различные **стили**, в частности, общий стиль формы *Form1* для всех компонентов. Он, однако, дает слишком большой шрифт, поэтому практически в *StyledSetting* отключают все компоненты стиля и в *TextSetting* устанавливаем заново для каждого компонента ему подходящий. Рекомендуется шрифт Tahoma 12 для обычного текста, 14—16 для заголовков и 10—11 для компонентов типа Таблиц, когда нужно загрузить большое количество данных. Свойство *WordWrap* (при этом должно быть свойство *AutoSize:=false*) – автоперенос на другую строку не вмещающегося текста следует определить как = *true*. Также часто нужно установить позицию текста = *center* (изначально *leading*).

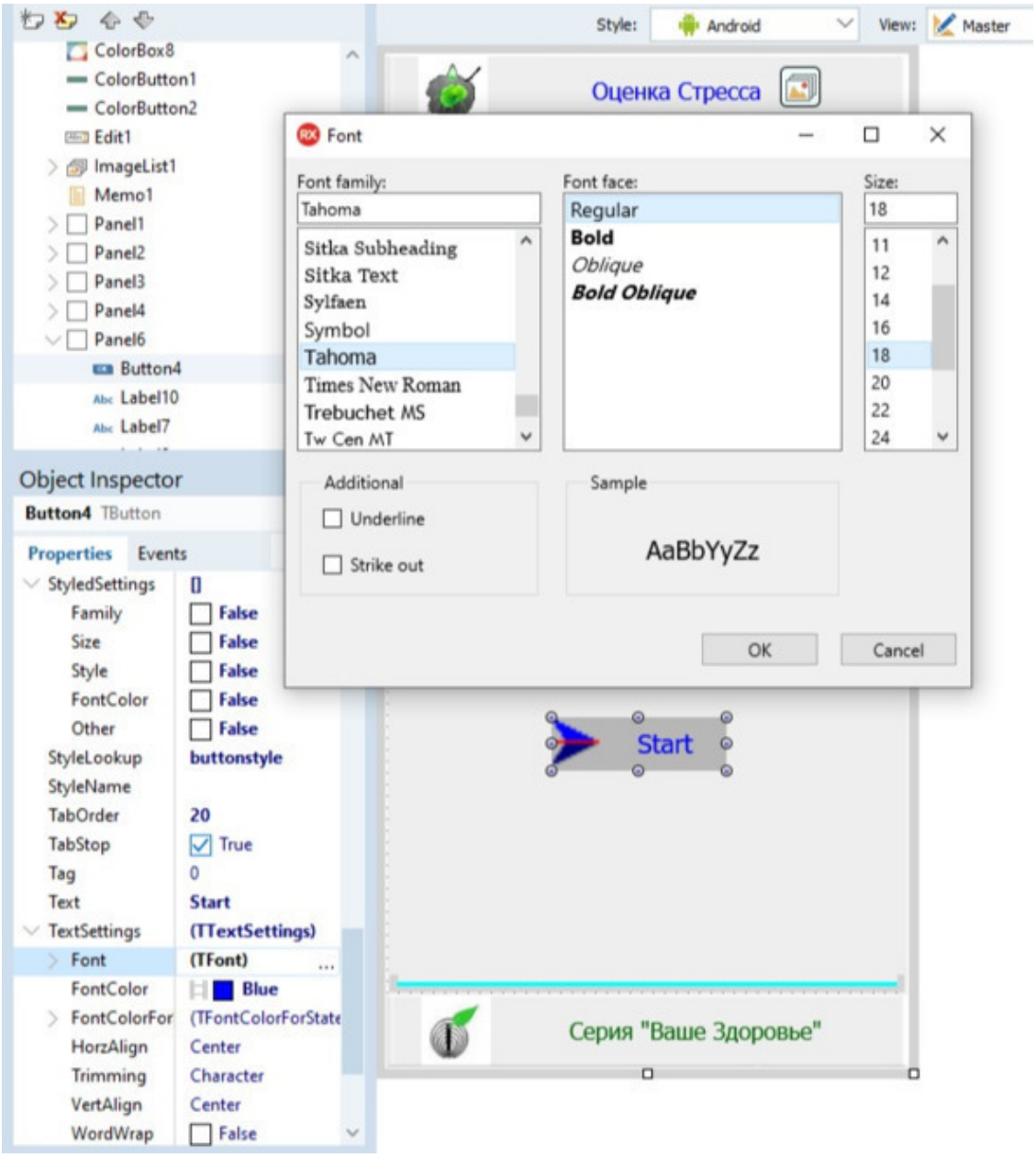


Рис. 12. Стили и их изменение.

Для компонентов имеется также свойство прозрачность: *Opacity,* по умолчанию 1.0 – полная непрозрачность компонента.

Для всех практически компонентов имеется свойство поворота на выбранное число градусов. Практически это свойство значения не имеет, кроме того, при повороте становятся невнятными свойства позиции и пр.

Еще одна проблема со смартфонами: *клавиатура* закрывает печатаемый текст. Для того, чтобы Видимость текста не заслонялась клавиатурой на сайте разработчиков можно найти модуль *vkbdhelper.pas*, и подключить к проекту: можно видеть, что пишите и в каком компоненте.

Еще один «подарок» от Delphi 10.3 – в режиме Android не доступны *Dialog*, поэтому обращение к файлам и пр. возможно только путем кода.

Крайне неудобно использовать Panel: она оказывается практически не видна и фактически может использоваться только как контейнер для различных компонентов на ней; она не может изменять цвет, и текст на ней отсутствует. Использование *Effects* позволяет полу-

чить бордюр (только выпирая панель вверх), но это фактически и все. Основное применение Панели – содержать компоненты в себе и выравниваться в Форме (хотя выравнивание вплоть до *Client* доступно и другим компонентам). Нагрузка Панели различными Effects кроме всего прочего не только утяжеляет программу, но иногда ведет к непредсказуемым влияниям на ее элементы.

3. Общие свойства компонентов

В Delphi 10.3 больше свойств для компонентов, чем в Delphi 7, но на практике оказывается, что наиболее важные у наиболее используемых компонентов отсутствуют, что проявляется в неприглядном виде Кнопок, невидимости Панелей и Окошек и пр. Авторы языка FireMonkey посчитали, что первичное плоское изображение компонентов нужно изменить в соответствии с необходимым дополнительными компонентами, доступными в группе *Effects*, в которой располагаются десятки компонентов-свойств.

На деле, практически все компоненты группы *Effects* предназначены для специальных эффектов типа мерцания, медленного исчезновения, жалюзи, скручивания и других мало пригодных для прикладных программ эффектов (а для разработки анимации и игр Delphi 10.3 все еще недостаточно эффективен и медленно работающий). Лишь некоторые из эффектов доступны и эффективны для обычных компонентов: *TBevelEffect* – эффект фаски (оптимально для Panel) и *TInnerGlowEffect* – внутреннее свечение, возможно: *TReflectionEffect* — отражение, *TGlowEffect* – наружное свечение, *TShadowEffect* – тень и *TBlurEffect* – размытие. Для задействования эффекта нужно перетащить компонент на форму и через Structure «подчинить» эффект визуальному компоненту управления.

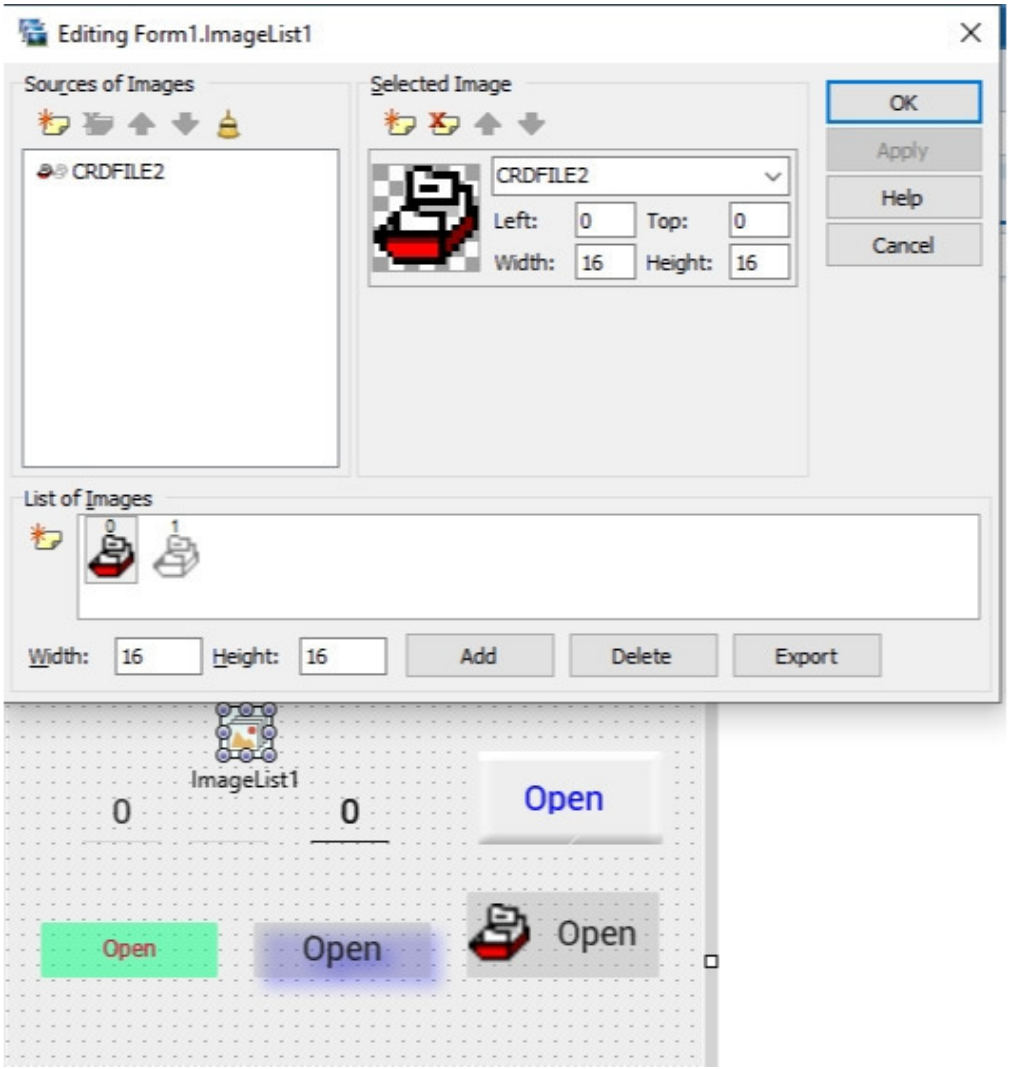


Рис. 13. Стили компонентов и добавление иконки через ImageList Editor.

На примере Кнопки видны возможности визуального отображения компонентов: изменение цвета компонента и текста, размера компонента и шрифта, размытие (тень) и добавление иконки (через *ImageListEditor*, при этом большая и расположенная с краю иконка неприглядна, увеличивает размер кнопки и практически для смартфона не применима). Видно также, что все эти художества не спасают компонент Кнопку и достичь обычного для Delphi вида не удается. Другие типы Кнопки из других групп также ничем не лучше. Получить красивую Кнопку можно из компонента Панель, задействовав *TBevelEffect* для получения объемного вида, но другие эффекты не воспроизводятся, как и не меняется цвет панели; к тому же компонент Панель утратила текст на ней (!) и для получения названия новоявленной Кнопки нужно сбросить на нее компонент *Label*. Все это резко утяжеляет программу, замедляет ее и иногда вообще отменяет написанный под компонент код.

Другой пример: Окошко (*Edit*): если в нем нет текста, то компонент практически не виден (как на рисунке 13 сразу под *ImageList)*, поэтому приходится сбрасывать на нее *TInnerGlowEffect,* который проявляется как уже ясно видимое подчеркивание.

Общие свойства компонентов видны из рисунка выше. Из множества свойств нужно менять лишь некоторые, остальные желательно не трогать.

Для компонентов обычно доступны:

– наиболее важное свойство: *Align* – выравнивание; в Android на смартфоне важно, чтобы компоненты не «ехали» на разных экранах. Из множества типов выравнивания общеприменимы: Вверх, Вниз, Вправо, Влево и *Client*, растягивающий компонент на весь контейнер (Панель, Форму и пр.), обычно используют несколько панелей вплотную и выравнивание Вверх и Вниз, а между ними – *Client*. Также имеется выравнивание *Scale*, впрочем, изменение размеров происходит автоматически для разных экранов.

– изменение размеров (и *auto Size* для *Label*),

– изменение цвета, в том числе градиентная заливка,

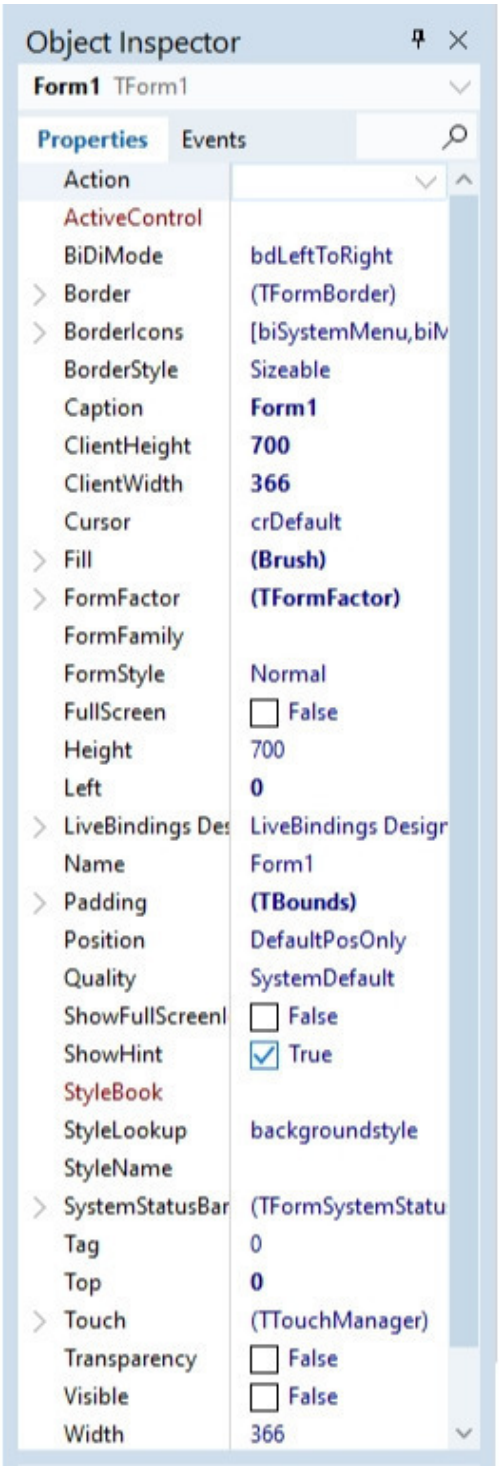


Рис. 14. Инспектор свойств компонентов.

– изменение размера, цвета и типа шрифта (часто центрируют его, так как изначально он расположен с краю – *leading*). Также для текстовых компонентов имеется свойство *WordWrap* (установите как *true*) для автоматического переноса на следующую строку не умещающегося текста; для просмотра не умещающегося на компоненте текста предусмотрено свойство *ShowScrollBars* и *EnabledScroll = true,* а также соответствующий компонент для скроллинга, что, впрочем, часто не работает, а текст видимый на экране при программировании, не помещается на экране смартфона (поэтому нужно часто просматривать текущий результат программирования на реальном подключенном устройстве, а не на виртуальных моделях),

– добавление иконки (через *ImageList Editor*), без возможности ее сдвига,

– видимость и доступность (*Visible* и *Enable* = true/false), причем иногда «недоступная» кнопка продолжает работать,

– возможность фокусируемости на данный компонент и «*Read only*»,

– стиль (обычно приходиться изменять установленный для всей формы стиль на подходящий для данного компонента),

– свойства *Position* (X и Y) от края контейнера (Панели, Формы) и отступы *Margin* – справ, слева, сверху и снизу,

– практически всем компонентам доступны вращение и анимации, однако, практически это нужно лишь для игр и фактически не используется.

4. Типы данных в Delphi 10.3

Типы данных, как обычно, **объявляются** перед началом программы (до *begin*) используя инициацию *var*:

```
var
x,y:Integer;
m: Double;
n: Float;
str1:String; // Строковая переменная.
```

```
Mas: array [1…100] of String; // Массив одномерный из 100 строк, начало с 0
Ar: array [0..9] of array [0..9] integer; // Многомерный массив, начало с 0.
D: array of real; // Динамический массив
MyChar: Char; // Тип для хранения простого символа.
R: TRect; //Область, ограниченную R. Left, R. Up, R. Right, R.Down.
```

Численные значения: Word, Integer, Doable, Float, Real; учитывая, что точность составляет 5—6 знаков, обычно используют *Integer* для целочисленных и *Double* для чисел с запятой, которые могут быть представлены также в формате E: 3.14E+2 = 3.14E+00 = 3.14; 23.5E-2 = 0.235. Для **округления** числа «x» удобно использовать *Round* (x), округляющий до ближайшего целого; также можно использовать для выделения целой части *Trunk* (x) и дробной части *Frac* (x).

Для ряда функций нужно добавить пакет **math** в User начала программы. Доступны многие математические функции, в том числе: абсолютное значение abs (x), квадрат sqr (x) и корень квадратный из x: sqrt (x); для степенной функции преобразование: x^n = exp (n*ln (x); корень n-й степени из x = exp (1/n*lnx). Имеется также функция возведения X^Y: *Power (x,y)*:

```
var
  Z: Real;
begin
  Z:= Power (0.25, 0.5); {Z:= 0.5}
end;
```

Функция получения **вероятного числа**: *Random* (n), по умолчанию пустые скобки () – вероятное число от 0 до 1; при «n» целочисленном – целочисленное значение от 0 до «n-1».

Символы представлены типом **Char**: Type: Char = #0..#255; *Char:= «3»*. Код символа можно узнать по функции *Chr (n)*. Код ANSI: #0…255. UNICODE: первые 256 символов = ANSI. Chr (66) = B; Char (67) = C.

```
var
  myChar: Char;
begin
   myChar:= «G»; // Назначение из символьной константы
    ShowMessage («Символ G = ' + myChar); //Получаем «Символ
G = G»
```

```
    myChar:= #65; // Назначение из целочисленной константы
    ShowMessage («#65 = ' + myChar); // Получаем «#65 = A»
end;
```

Присваивание значения требует двоеточия перед равенством: *n:= 10;* обычное равенство (=) используется в булевых значениях сравнения (true/false): ***if*** *n = 10* ***then***… Неравенство: X <> Y.

Массивы: важная часть программ для накопления данных. Могут быть:

– одномерные: *Mas: array [1…100] of String*; // все начинаются с 0.

– многомерные: *Ar: array [0..9] of array [0..9] of integer*;

– динамический массив: *D: array of real*.

Перед использованием динамического массива устанавливается его длина (начинается с 0): *setLeangth (D,20)*. Закрытие массива: *D_M: =nil*. элементы массива начинаются с 0, исключая **строковой массив**, начинающийся с 1.

Доступ к массиву по его индексу: n:= D [21]; соответственно: D [21]:= n. Для заполнения массива обычно применяется конструкция цикла:

```
for n:= 1 to 100 do
begin
D [n]:= n-1; //Нумерует компоненты массива D [100], начиная с 0
end;
```

Можно найти максимальное и минимальное значение и среднее по массиву: *MaxIntVal (D)*; *MinVal (D): double); Mean (D):double)*, копировать массив в другой с определенного компонента: *D:= copy (D, 0, 20)* и др.

Строки: *ShortString*: 255 символов и занимает в памяти 2 байта; String = AnsiString: 1031 символ. *AnsiString* или *WideString* содержат большое количество символов. В типе *AnsiString* символы кодируются в коде ANSI, а в типе *WideString* в коде Unicode. Общим типом является тип String. String [n] – ограничивает длину строки.

Со строками можно проводить множество операций:

– найти в строке субстроку, ее индекс: *n:= Pos (subStr, str)*;

– копировать строку str с позиции index и числом знаков count: *str1:= copy (str, index, count);*

– аналогично удалить часть строки по индексу *delete (str, index, count)*;

– вставить субстроку: *insert (str1, str0, index);*

– длину строки можно узнать как: *length (str).*

Для выравнивания строк полезна функция вставки пустых символов: *StringOfChar (»», count);* если использовать моноширинный шрифт, то получим идеально выровненные в столбик строчки.

SelStart и *SelLength* устанавливают позицию 1-го символа и длину записи, *SelText* – выделенный текст, весь текст выделяет *SelectAll; ClearSelection* очищает выделенный текст; *Clear* очищает текст, *Undo* возвращает предыдущее действие.

Для копирования текста используют *CopyToClipBorad*, для вставки *Paste*, для замены символов *StringReplace* (). Перед копированием целесообразно выделить текст для копирования; так, *Memo.CopyToClipBorad* ничего не копирует, для копирования всего содержимого компонента Memo нужно предварительно выделить его содержимое:

```
Memo.SelectAll;
Memo.CopyToClipBorad
```

Перевод каретки: *string1* + ***#13#10*** + *string2* (перевод каретки и конец строки с переносом string2 на другую строку).

Изменение типов проводится очень часто, обычно для передачи строкового значения в числовое и наоборот, используют: *StrToInt*; *InToStr*; *StrToFloat* (str); для форматирования: *FloatToStrF* (*n*, *ffGeneral* или *ffFixed*, *count* знаков всего, *count* знаков после зпт). Вместо *Float* используют *Double*.

Дата и **Время**. Имеется значительное число возможностей работы с датой, но обычно достаточно узнать настоящую дату *DateToStr* (Now): и текущее время: *DateTimeToStr* (Now).

```
Форматы представления даты:
dd/mm/yy hh: mm: ss = 09/02/49 01:02:03
mmm = Feb
mmmm = February
ddd = Tue
dddd = Tuesday
ddddd = 09/02/2049
dddddd = 09 February 2049
hhampm = 01AM
t = 01:02
tt = 01:02:03
dd/mm/yyyy = 09/02/2049
dd/mm/yy hh: mm: ss = 09-02-49 01_02_03
mmm = FEB
mmmm = FEBRUARY
ddd = WED
dddd = WEDNESDAY
ddddd = 09-FEB-49
dddddd = WEDNESDAY 09 of FEBRUARY of 1949
hhampm = 01morning
t = 01_02_03
tt = 01 _ 02 _ 03. 004
dd/mm/yyyy = 09-02-1949
```

Для отсчета времени в Delphi 10.3 имеется такой же компонент, как и ранее: *Timer*, запускающийся при присвоении функции *Timer*. *Enable*:= *true* и выключаемый присвоением *…false*; имеет единственное действие, повторяемое через задаваемый промежуток времени (в мСек). Однако, кроме малой точности (50 мСек) при достаточно нагруженной программе он действует крайне медленно, может тормозить в 2 раза и несколько секунд включаться, так что практически мало пригоден (хотя вне Android работает вполне точно и хорошо). Для получения времени между 2-мя событиями в Delphi 10.3 удобнее всего использовать функции даты-времени:

Разница 2-х времен в Сек:

```
// На Кнопку Start
var
str, h, m, s: String;
t1:Integer;
```

```
begin
str:= DateTimeToStr (Now); // Начальное время
h:= Copy (str, 12, 2);// Отсекаем Дату и сразу берем значение Часы
m:= Copy (str, 15, 2);
s:= Copy (str, 18, 2);
t1:= StrToInt (h) *3600 + StrToInt (m) *60+ StrToInt (s); //Текущее время в сек
Edit1.Tex:= IntToStr (t1);
end;
// На Кнопку Stop
var
str, h, m, s: String;
t, t1, t2:Integer;
begin
str:= DateTimeToStr (Now); // Конечное время
h:= Copy (str,12,2);
m:= Copy (str,15,2);
s:= Copy (str,18,2);
t2:= StrToInt (h) *3600 + StrToInt (m) *60+ StrToInt (s);
t1:= StrToInt (Edit1.text);
t:= t2 – t1;
Edit3.Text:= IntToStr (t); // Итоговое прошедшее время
end;
```

Можно также использовать функцию разбора времени на составляющие:

```
var
Hour, Min, Sec, Msec: Word;
begin
DecodeTime (Now, Hour, Min, Sec, Msec);
Edit1.Text:=IntToStr (Hour) +IntToStr (Min) +IntToStr (Sec) + IntToStr (Msec);
end;
```

5. Вкладки компонентов

Базовые компоненты находятся на вкладке **Standard**, вкладка **Additional** используется значительно меньше, так как ее элементы практически дублируют вкладку Standart; из вкладки **System** можно использовать компонент *Timer* (с отмеченными ограничениями), вкладка ***Win 32*** исчезла.

Компонент Таблица представлен во вкладке **Grids** двумя таблицами – обычной строковой *StringGrid* и *Grid* в котором можно использовать самые разные типы данных и вставлять другие компоненты, также отдельный компонент Заглавие – *Header*.

Желание разнообразить возможности рисования обернулись их резким усложнением, однако, появилась вкладка **Shapes**, где представлены различными компонентами: прямоугольник, круг, овал, дуга, линия, подпись и другие типичные элементы рисунков, свойства которых можно задавать программно и рисовать или создавать графики.

Компоненты графиков представлены единственным *PlotGrid* с сеткой и осями координат, на котором фактически нужно рисовать график вручную.

Вкладку **Effects**, задуманная для придания красивого вида элементов, использовать можно разве что для того, чтобы сделать хоть как-то видимым компонент *Edit*. Различные эффекты мигания, скручивания, жалюзи и пр., как и ротация, доступная практически для всех элементов, предназначены для анимации (как и вкладка **Animations**, вкладка **Styles** и вкладки для **3D** вида компонентов), к которой сам Delphi как раз не предназначен, и работать на смартфоне будет в лучшем случае очень медленно. Для прикладных программ это вообще дурной тон.

Видимо, вкладка **LiveBinding**, нововведение авторов языка Firemonkey, отслеживающая «жизнь» компонентов и их связь между собой, также вряд ли подойдет для привыкших к обычному программированию.

Вкладка **Color** представлена множеством компонентов для работы с цветом, часто взаимодействующими между собой, и также скорее подходят для анимации, чем для прикладных программ. Определенный интерес представляет возможно простой *ColorBox*. Для создания цветового градиента с целью приданию графику вида «зеленая и красная области» подойдут два компонента *Rectangle* (прямоугольник) из вкладки *Shapes*, на которые накладываются другие прямоугольники уже как графики данных.

Как уже отмечалось, важными элементами приложения являются компоненты их вкладки Layouts, отвечающие за выравнивание компонентов, в них находящихся, хотя со времен первых вариантов Delphi эту функцию успешно играет компонент Панель – *Panel*, которая, к сожалению, лишилась теперь свойств цвета и текста, как и элементов бордюра.

Вкладка **Common Control** представлена рабочими элементами – *TabControl* и *MultiView*.

Важнейшая вкладка **Dialogs** в Delphi 10.3 отсутствует (вернее, не доступна для платформы Android).

Целый ряд вкладок предназначен для работы с Базами данных, в том числе **SQL** и специальная с широкими возможностями – **Fire**, вместе с вкладками для работы в **Интернет**.

Особое значение для Android имеют компоненты работы с элементами смартфона: камерой, сенсорами, аудиоплеером, интернетом, *Bluetooth* (вкладка *Systems*), что, впрочем, гораздо лучше делает сам смартфон. Для работы нужно включить флажок в: *Project – Options – Uses Permissions*.

Важнейшей, однако, остается вкладка Standard, в которой сосредоточены компоненты ввода и вывода текста и цифровых данных и действий над ними.

Для обычной рутинной работы для каждого приложения понадобятся:

– из вкладки **Standard**: *Label* (метка), *Edit* (окошко), *Memo* (многостраничный текстовый компонент, аналог отсутствующего Rich Edit), *Button* (кнопка), *CheckBox* или *RadioButton* для выбора условий, *ListBox* (списки), *ImageControl* вместе с *ImageList* Panel для ввода иконок и картинок;

– из вкладки **Addinional**: *PlotGrid* для графиков, *NumberBox* для ввода числовых значений, *ComboEdit* для сложного ввода, и возможно *MediaPlayer*;

– из вкладки **Grids**: обычная *StringGrid* для текстовых данных;

– вкладка **Shapes**: с уже готовыми фигурами неоценима для создания графиков;

– из вкладки **Colors**: возможно, *ColorBox*;

– из вкладки **Effects**: *BevelEffect* для создания бордюра компонента Panel, и *TinnerGlowEffect* для придания видимости компоненту Edit;

– компоненты вкладки **Layouts** нужны для выравнивания содержащихся других компонентов, хотя традиционно эту роль выполняет компонент Panel.

Таким образом, основные рабочие компоненты сосредоточены на вкладке **Standart** и большинство предназначено для ввода/вывода информации, причем текстового характера.

Типичные компоненты для ввода текста:

– Метка – *Label*,

– Окошко – *Edit*,

– компонент многостраничного ввода – *Memo* (RichEdit отсутствует),

– Списки – *List*.

Свойства компонентов можно задавать в Инспекторе объектов или кодом, соответствующим свойствам. При добавлении на Форму компонент появляется в соответствующем Форме стиле, поэтому приходится для каждого компонента убирать все элементы стиля (ставить *false* во всех свойствах *StyledSettings*) и создавать свой собственный стиль, нужный именно этому компоненту; также можно установить видимый стиль компонента (*StyleLookup*), обычно его можно пропустить или ставить всегда соответствующим данному компоненты.

Общие свойства (размеры, стиль, положение, видимость, доступность, фокусируемость, позиция и отступы, свойство «только для чтения», свойства шрифта и пр.) описаны выше, также имеются некоторые собственные свойства и способы работы с компонентами.

6. Компоненты для ввода и вывода текста

6.1. Компонент Метка – Label

Компонент **Label** – предназначен для текстовой информации, например, заголовков, но подойдет и для ввода/вывода любого короткого текста. При использовании как заголовка ему лучше придать значение выравнивание *Align*: *Top* или *Bottom* и растянуть на всю Форму, не забыв центрировать текст в *TextSetting* (изначально свойство представлено как *leading*).

Общие *StyledSetting* обычно приходится убирать (false) и ставить собственные в *TextSettings,* свойства ротации и анимации вряд ли нужны.

Важны также свойства: *ReadOnly* (запрещает вход и изменение текста, хотя программно можно изменять текст) и *AutoSize* – автоматический размер под текст, который можно и не ставить, а лучше сделать побольше размер в высоту и ширину.

Также важно свойство *WordWrap = true* в *TextSetting*, которое автоматически переводит не вмещающийся текст на строчку ниже.

Свойства компонента можно задавать и кодом. Работа с компонентом состоит обычно в передаче и считывании текста:

```
Label1.Text:= «Вводимый текст»; //текст в кавычках
Label1.Text:= str; // str – строковая переменная
str:= Label1.Text;
Label1.Text. Empty = «»;
```

Начальный текст обычно задается в Инспекторе объектов, как и его характеристики, которые можно поменять программно в ходе работы приложения.

Из длинного списка свойств и действий, предлагаемого подсказкой после ввода названия компонента и точки, означающей ввод свойств или действий, на самом деле используется обычно только ввод и вывод текста, а чаще всего Label используется вообще однократно при создании приложения для заголовков или подписей.

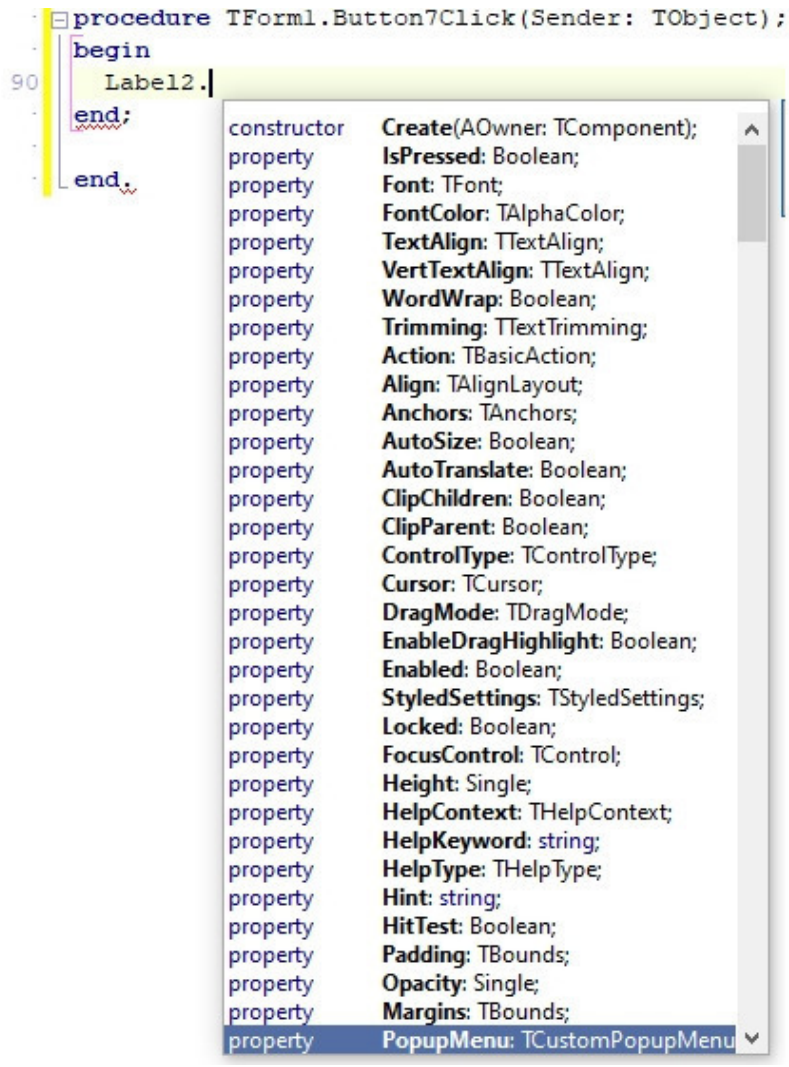


Рис. 15. Предлагаемые программно свойства и действия для компонента Label.

6.2. Компонент Окошко – *Edit*.

Компонент *Edit*, наверно, наиболее используемый в работе (наряду с Кнопкой – *Button*, которая и задает действия с ним).

Если в обычном Delphi это обычное белое окошко, то в Delphi 10.3 это фактически не видимый элемент, с едва заметным подчеркиванием, которое приходится доводить до видимости компонентом из вкладки *Effects*

Конец ознакомительного фрагмента.