

Міністерство освіти і науки України
Донбаська державна машинобудівна академія (ДДМА)

ПРИНЦИПИ ПОБУДОВИ ІНТЕРФЕЙСІВ МОБІЛЬНИХ СИСТЕМ

РОЗРОБКА МОБІЛЬНИХ ДОДАТКІВ У СЕРЕДОВИЩІ EMBARCADERO DELPHI

Методичні вказівки
до виконання лабораторних робіт
студентів спеціальності 122
«Комп'ютерні науки та інформаційні технології»
всіх форм навчання

Затверджено
на засіданні
методичної ради
Протокол № від

Краматорськ
ДДМА
2017

Принципи побудови інтерфейсів мобільних систем. Розробка мобільних додатків у середовищі Embarcadero Delphi : методичні вказівки до виконання лабораторних робіт студ. спец. 122 «Комп'ютерні науки та інформаційні технології» всіх форм навч. / уклад.: А. В. Бабаш – Краматорськ : ДДМА, 2017. – 74 с.

Наведено опис інтегрованого середовища розробки сучасних додатків Embarcadero Delphi. Представлено його опис та основні можливості для розробки мобільних додатків операційної системи Android. Викладений теоретичний матеріал, який може бути корисним при виконанні циклу лабораторних робіт. Також наведений повний програмний код додатків, які представлені у якості прикладу для виконання лабораторних робіт студентами.

Методичні вказівки призначені для студентів при виконанні лабораторних та практичних робіт при вивченні дисципліни "Принципи побудови інтерфейсів мобільних систем".

Відп. за випуск

А. Ф. Тарасов, д.т.н., проф., зав. каф. КІТ

Навчальне видання

ПРИНЦИПИ ПОБУДОВИ ІНТЕРФЕЙСІВ
МОБІЛЬНИХ СИСТЕМ

РОЗРОБКА МОБІЛЬНИХ ДОДАТКІВ У
СЕРЕДОВИЩІ EMBARCADERO DELPHI

Методичні вказівки до самостійної роботи
студентів спеціальності 122 «Комп'ютерні науки та інформаційні технології»
всіх форм навчання

Укладачі: БАБАШ Андрій Владиславович,

За авторським редагуванням

Комп'ютерне верстання

О. М. Болкова

100/2016. Формат 60 × 84/16. Ум. друк. арк. 1,09. Обл.-вид. арк. 0,93.

Тираж пр. Зам. №

Видавець і виготівник Донбаська державна машинобудівна академія
84313, м. Краматорськ, вул. Шкадінова, 72.

Свідоцтво суб'єкта видавничої справи ДК №1633 від 24.12.2003

ЗМІСТ

Вступ.....	4
1 Лабораторна робота №1.....	4
2 Лабораторна робота №2.....	28
3 Лабораторна робота №3.....	42
4 Лабораторна робота №4.....	54
5 Лабораторна робота №5.....	64
6 Лабораторна робота №6.....	70
7 Лабораторна робота №7.....	73
Висновки	88
Перелік посилань.....	89

ВСТУП

Embarcadero Delphi являє собою RAD (rapid development) середовище швидкої розробки додатків. Воно включає багато різноманітних компонентів, які суттєво підвищують швидкість розробки та дозволяють у максимально короткий час представити розроблений додаток замовнику. Середовище дозволяє візуально побудувати інтерфейс майбутнього додатку, що суттєво спрощує розробку.

Середовище Embarcadero Delphi дозволяє розробляти додатки для різноманітних сучасних популярних операційних систем, таких як MS Windows, Android, Mac Os, IOs, Ubuntu Linux на основі єдиної кодової бази. Тобто розробнику не потрібно буде розробляти окремий додаток для кожної операційної системи. Дане середовище дозволяє створювати нативні для додатки наведених операційних систем. Тобто для операційної системи Android буде створено Ark архів зі скомпільованими Java – класами. Скомпільований архів Ark можна встановити та запустити на мобільному пристрої з процесором ARmv7 (пристрої з відео картою Tegra не підтримуються).

Embarcadero Delphi являє собою кросплатформене середовище, яке дозволяє збирати додатки для різних операційних систем. Крім стандартних візуальних компонентів середовище дозволяє використовувати сторонні, які розширюють можливості розробки для різних платформ.

1 ЛАБОРАТОРНА РОБОТА №1

Тема. Ознайомлення з основними елементами для побудови інтерфейсу мобільного додатка. Робота з Ini файлами.

Мета. навчитися проектувати інтерфейс мобільного додатка та використовувати Ini файли для збереження і завантаження різних налаштувань мобільного додатка.

Хід роботи

Для розробки мобільних додатків використовується середовище швидкої розробки додатків Delphi XE8.

Після запуску і успішного завантаження Delphi XE8 з'являється наступний інтерфейс (рис. 1.1).

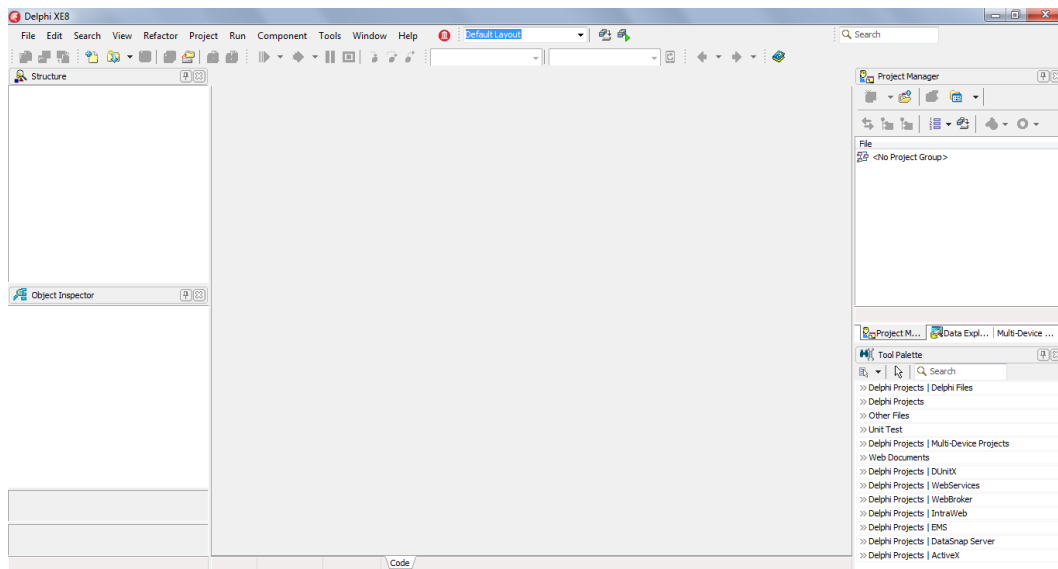


Рисунок 1.1 – Інтерфейс середовища Delphi при завантаженні

Для створення мобільного застосування вибираємо File -> New -> Multi-Device Application - Delphi. Далі обирається Blank Application (рис. 1.2).

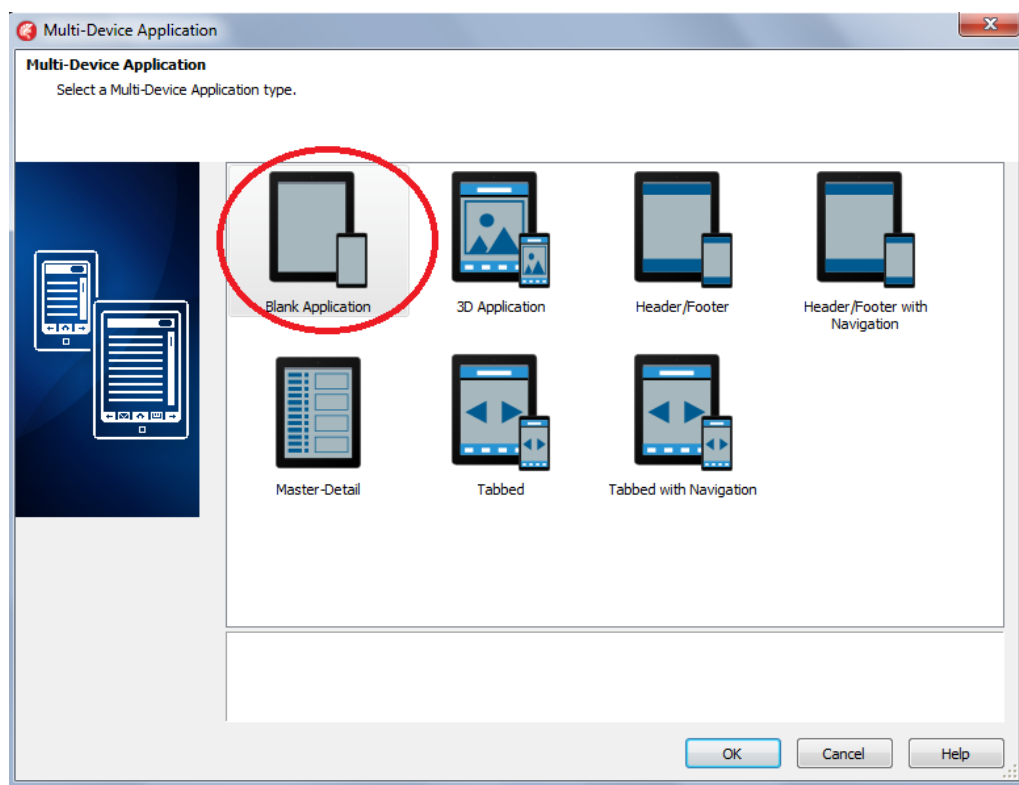


Рисунок 1.2 – Вибір шаблону кросплатформеного додатку

Середовище Delphi дозволяє створювати крос-платформні додатки на основі фреймворку FireMonkey [1,2].

Для створення додатків для Андроїд потрібно виконати наступні налаштування (рис. 1.3).

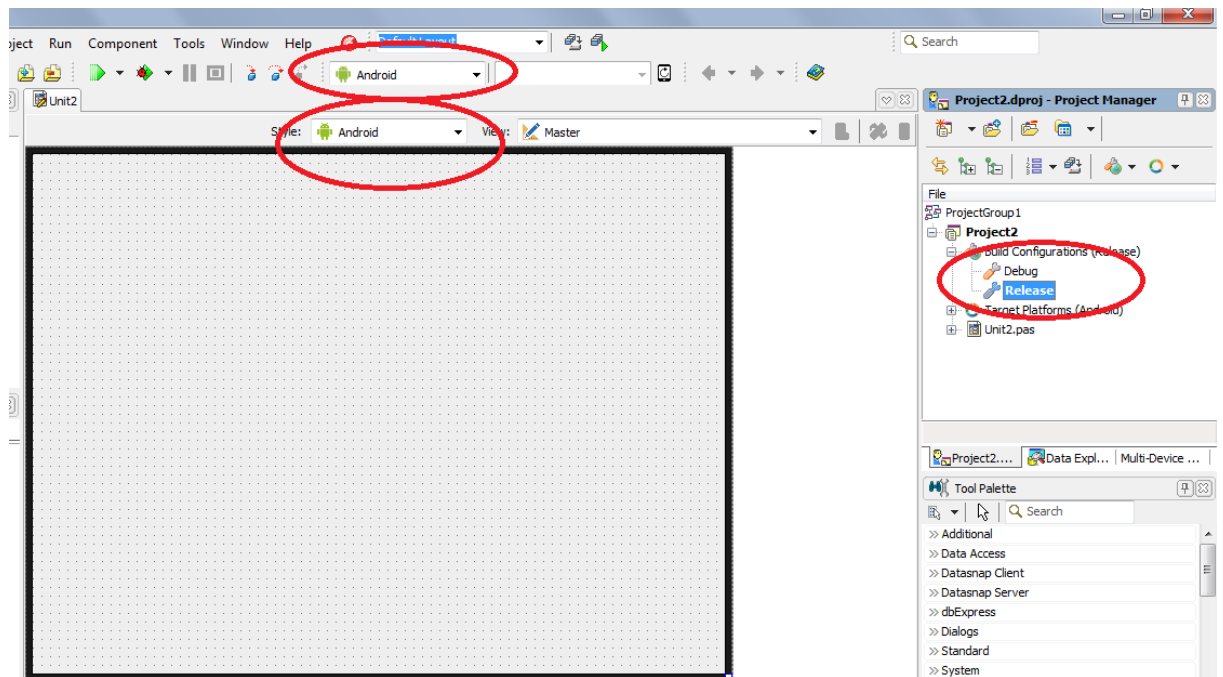


Рисунок 1.3 – Налаштування середовища для розробки на Android

Структура та зовнішній вигляд проекту Lab_1 у середовищі Embarcadero Delphi XE8 має наступний вигляд (рис. 1.4).

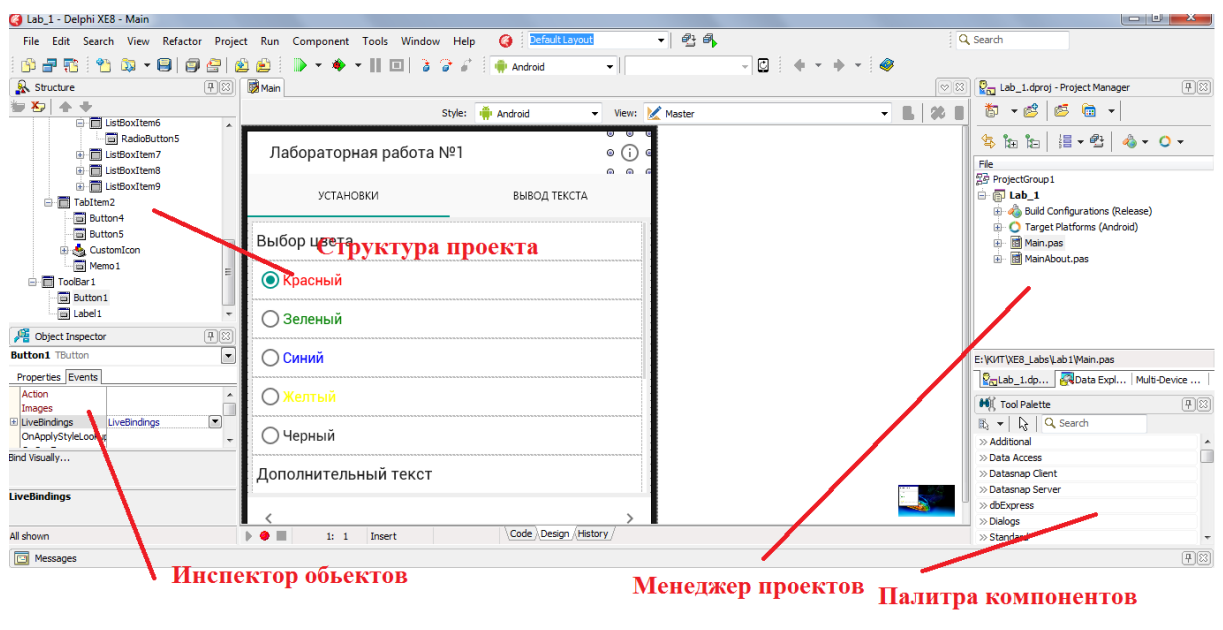


Рисунок 1.4 – Проект Lab_1

Инспектор об'єктів містить властивості (properties) і події (events) поточного об'єкта (головної форми, кнопок, полів введення і т.д.).

Палітра компонентів дозволяє вибрати потрібний компонент (об'єкт) і включити його в проект.

Вікно структури проекту містить в собі всю структуру розроблюваного проекту (всі обрані компоненти).

Менеджер проектів дозволяє перемикання між проектами. Також містить в собі конфігурацію проекту для його побудови і компіляції, включає в себе доступні емулятори як віртуальні так і фізичні (реальне андроїд пристрій - смартфон або планшетний ПК).

Палітра компонентів містить безліч різних компонентів для створення інтерфейсу і дозволяє їх додати на свою домашню форму простим перетягуванням (Drag and Drop). Однак можна створювати компоненти і динамічно (тобто написанням програмного коду).

Далі описаний механізм додавання компонента Button класу TButton з палітри інструментів Standard (рис. 1.5). При виборі збірки Release розмір андроїд-додатку (apk) скорочується.

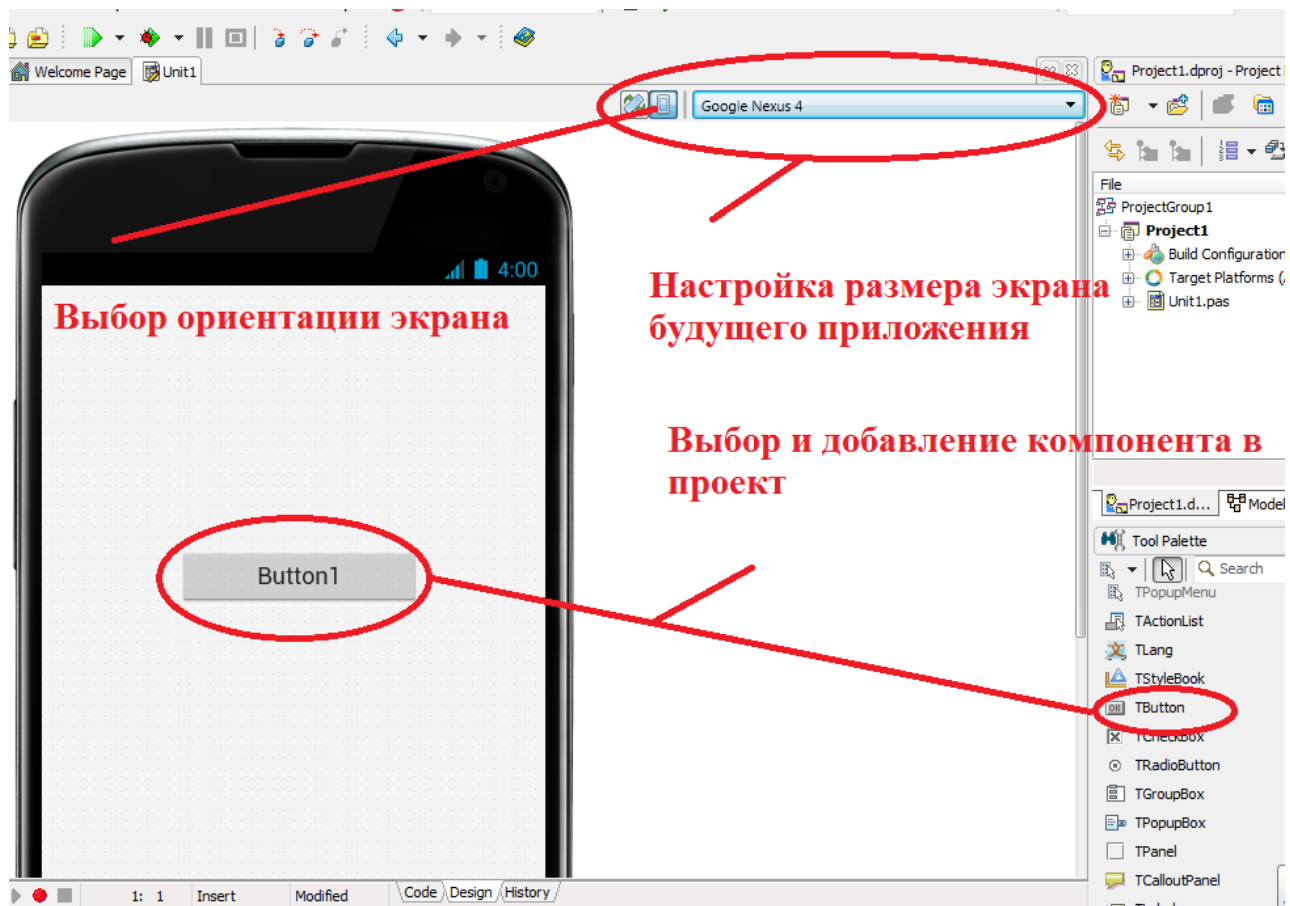


Рисунок 1.5 – Формування інтерфейсу мобільного додатка

Кожен компонент (об'єкт) має набір властивостей (Properties в інспекторі об'єктів) та подій (Events в інспекторі об'єктів). Для зміни напису кнопки Button1 класу TButton використовується властивість Text (рис. 1.6)



Рисунок 1.6 – Встановлення властивостей компонентів за допомогою Object Inspector

Кожен об'єкт має певний набір подій (OnClick, OnMouseMove і т.д.). Для створення обробників подій потрібно перейти на вкладку Events вікна Object Inspector. При подвійному натисканні на рядку події генерується заготовка його обробника, де власне і пишеться програмний код обробника даної події (рис. 1.7) [3].

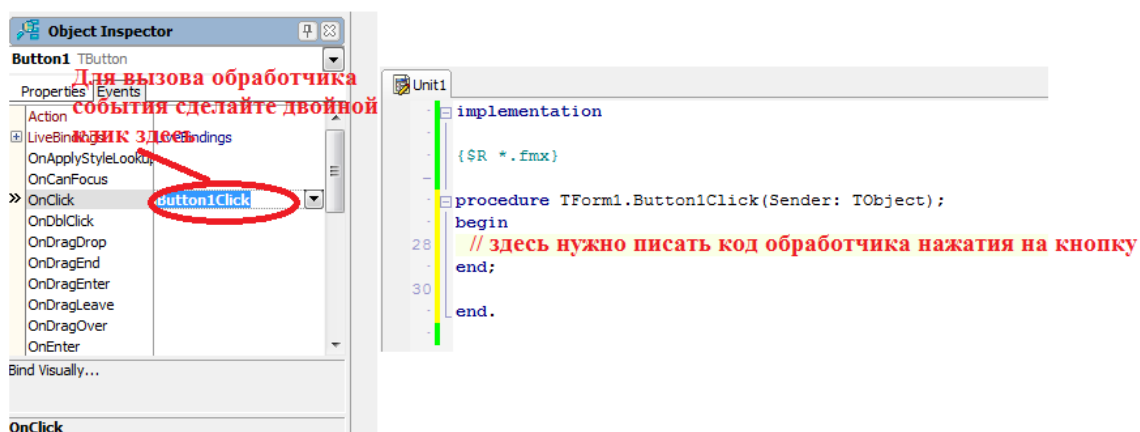


Рисунок 1.7 – Підключення обробника певної події

Для швидкого пошуку потрібного компонента по його назві в палітрі компонентів є рядок пошуку (рис. 1.8).

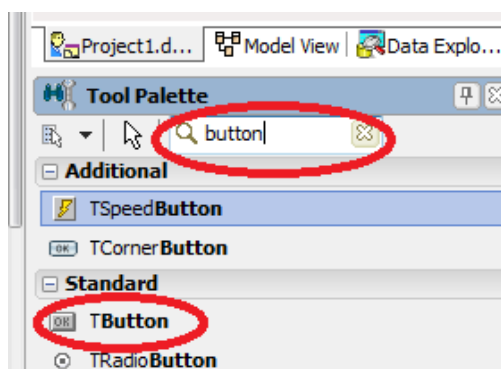


Рисунок 1.8 – Швидкий пошук компонентів

Порядок виконання лабораторної роботи

Практично всі андроїд – додатки вгорі форми мають панель інструментів. Вона додається на форму з палітри інструментів простим перетягуванням (рис. 1.9)

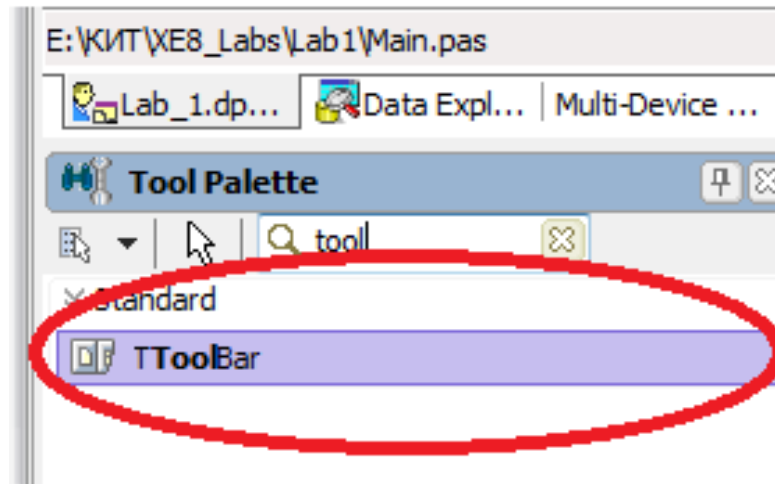


Рисунок 1.8 – Панель ToolBar

Для прив'язки панелі інструментів до верхньої частини форми потрібно встановити властивість Align -> Top [4] (можна проекспериментувати з цією властивістю). Для андроїд у цього компонента є ще властивість TintColor, тобто колір панелі інструментів (рис. 1.9).

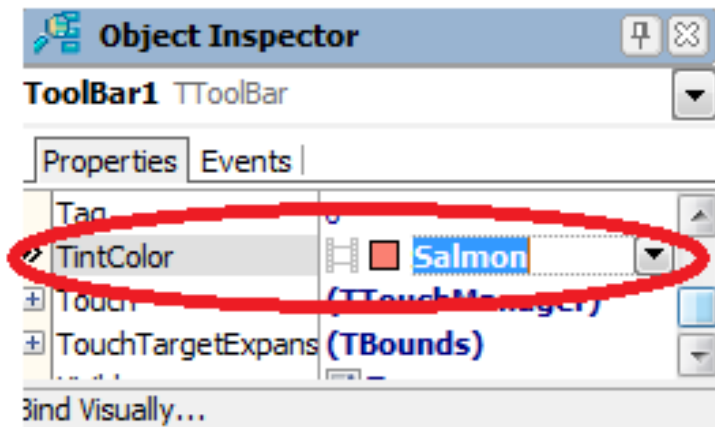


Рисунок 1.9 – Властивість TintColor

Далі аналогічно додається компонент Label і Button. Кожен компонент з палітри інструментів може бути контейнером для інших компонентів, що може бути дуже зручно при створенні різних складних інтерфейсів. Тобто кожен компонент може бути "батьком" (Parent) для інших компонентів, які включені в нього. Прикладом може служити вкладення Label і Button в ToolBar (рис. 1.10).

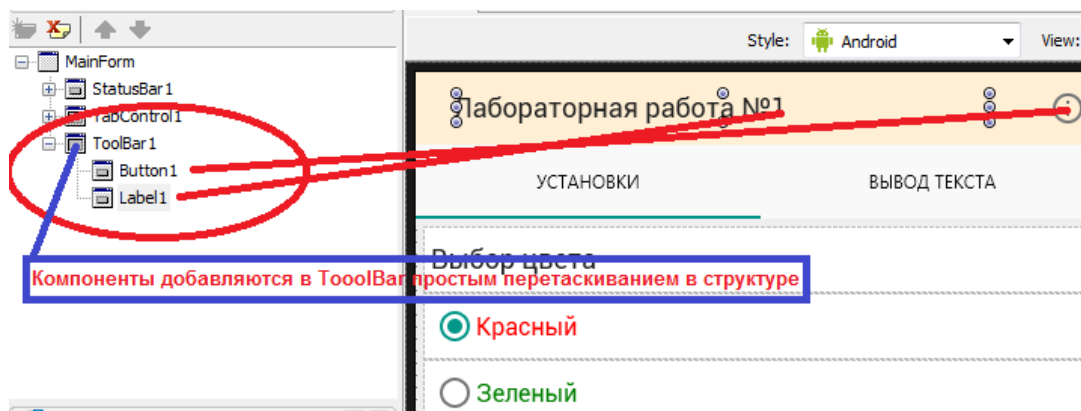


Рисунок 1.10 – Вкладення компонентів

Властивість Align компонентів Label і Button встановлюються в Center і Right відповідно.

Далі потрібно додати компонент TabControl (Align-> Client). Він дозволяє посторінково розмістити компоненти і також перемикається між сторінками. Для додавання сторінок потрібно натиснути правою клавішею миші на компоненті і вибрати пункт контекстного меню Add TTabItem [5,6].

Зауваження. У кожного компонента є властивість Name (тільки латиницею). Воно дозволяє задати ім'я кожного об'єкта, для звернення з програми. На практиці дуже неінформативні імена Button1, Tollbar1, Form1. Особливо, коли інтерфейс є складним. Можна назвати Button1 наприклад MyMainButton.

Після потрібно встановити назву кожної сторінки (TabItem властивість Text) (рис. 1.11).

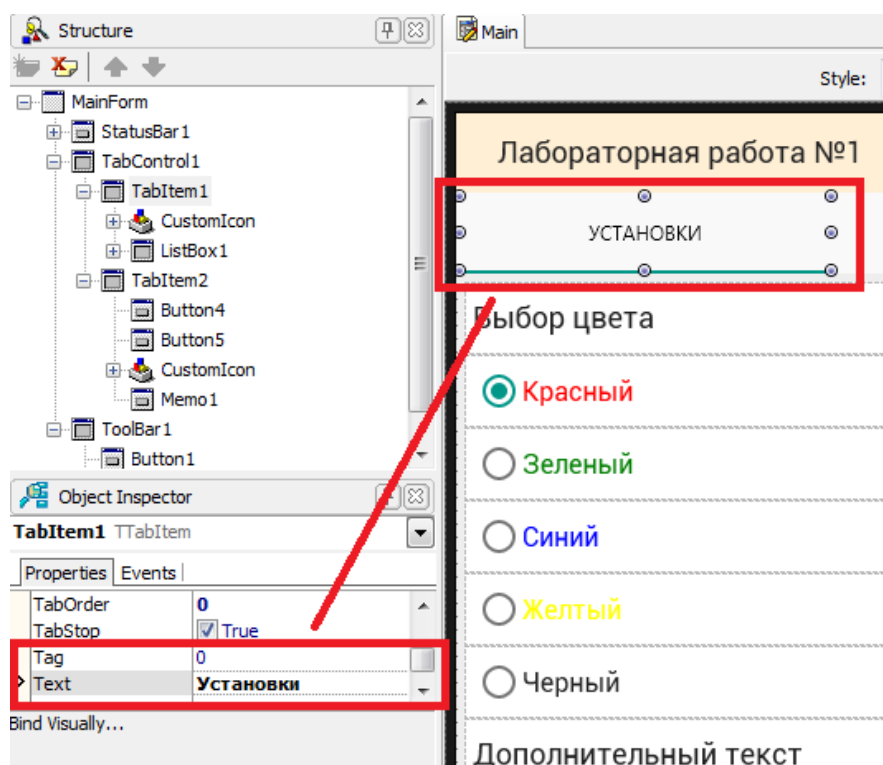


Рисунок 1.11 – Створення напису для TabItem

Для відображення перемикачів та інших елементів інтерфейсу у вигляді красивого і рівного списку використовується ListBox [7]. Він буде контейнером для перемикачів Radiobutton, текстових полів Edit, кнопок і т.д.

У кожного компонента є властивість Padding. Воно дозволяє встановити відступи компонента для його дочірніх елементів, тим самим дозволяючи домогтися гарних ефектів оформлення інтерфейсу (рис. 1.12).

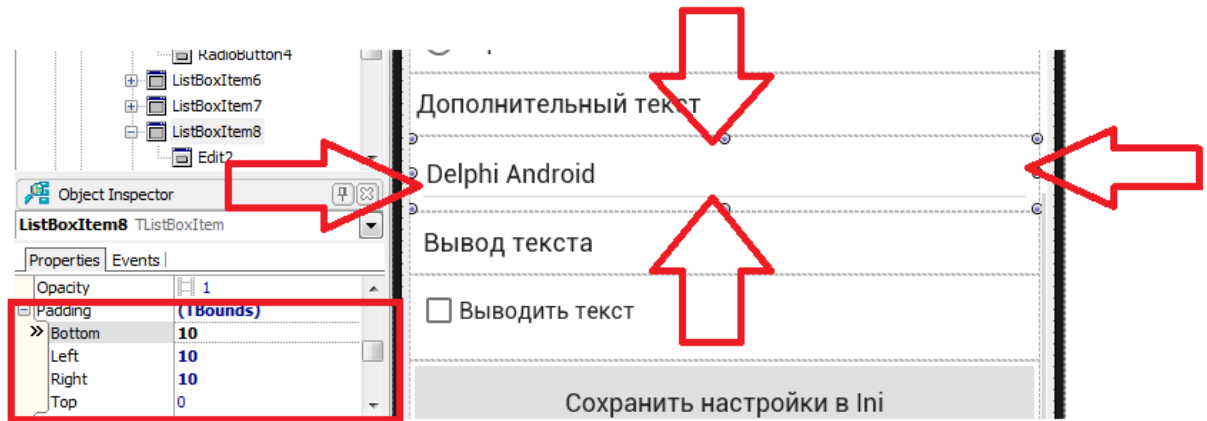


Рисунок 1.12 – Встановлення відступів

Компонент ListBox додається з палітри інструментів. Властивість Align - > Client, тобто розгорнути на всю доступну клієнтську область. (В даному випадку TabItem). Далі потрібно додати елементи списку ListBoxItem (правої клавищи по ListBox, вибір меню Items Editor) (рис. 1.13) [8]

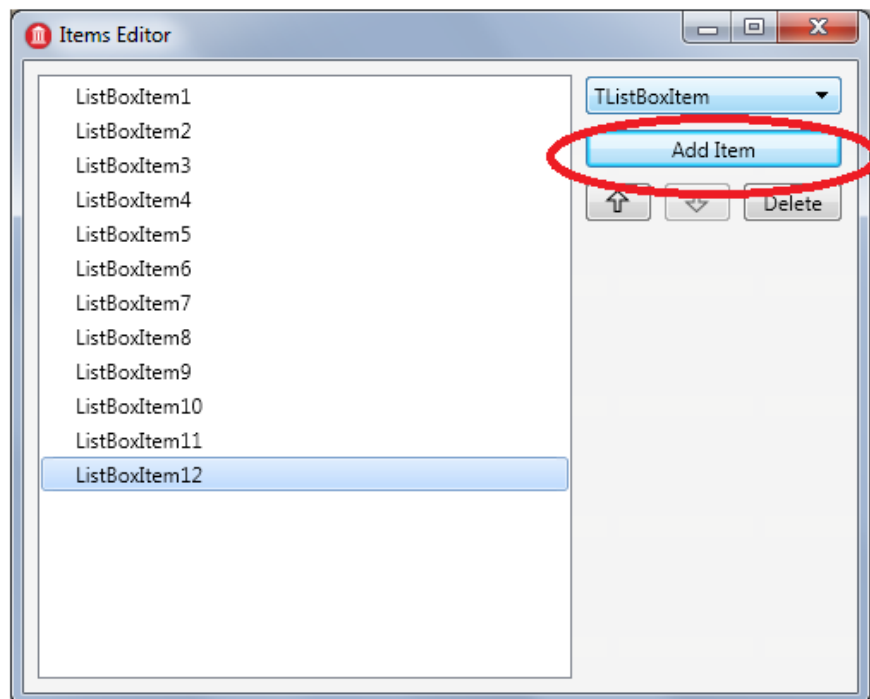


Рисунок 1.13 – Додавання елементів списку

Кожен ListBoxItem може виступати контейнером для інших об'єктів (RadioButton, CheckBox, Edit, Button).

ListBoxItem як і будь-який інший компонент має властивість Text. У нашому випадку потрібно встановити властивість Text всіх ListBoxItem в ".

Компонент RadioButton, CheckBox мають важливі властивості IsChecked, які дозволяють вибрати перемикач (рис. 1.14).

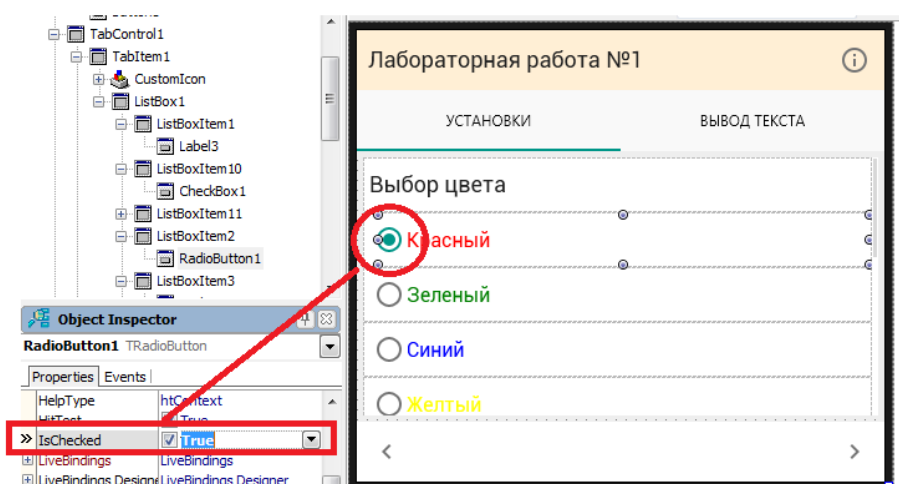


Рисунок 1.14 – Встановлення властивості IsChecked

Для кожного перемикача можна вибрати колір і тип шрифту (рис. 1.15).

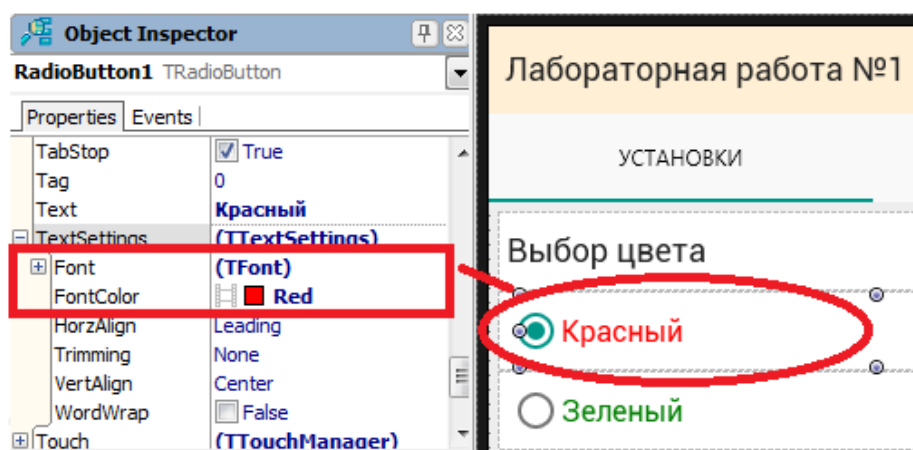


Рисунок 1.15 – Встановлення шрифту

Далі потрібно додати на форму StatusBar (його контейнером буде головна форма). У StatusBar потрібно додати дві кнопки Button для перегортання сторінок TabControl. Властивості Align кнопок повинно бути Left і Right відповідно, щоб розташувати їх по лівому і правому краю StatusBar відповідно (рис. 1.16).

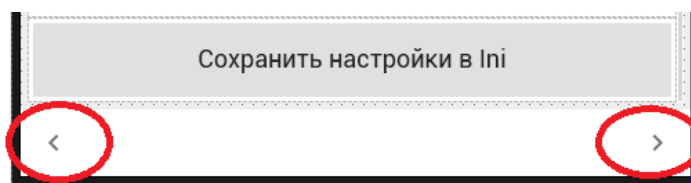


Рисунок 1.16 – Перемикання TabControl

Для завдання стилю кнопок потрібно встановити властивість StyleLookup (з цією властивістю можна експериментувати, вибираючи різні його значення) (рис. 1.17).

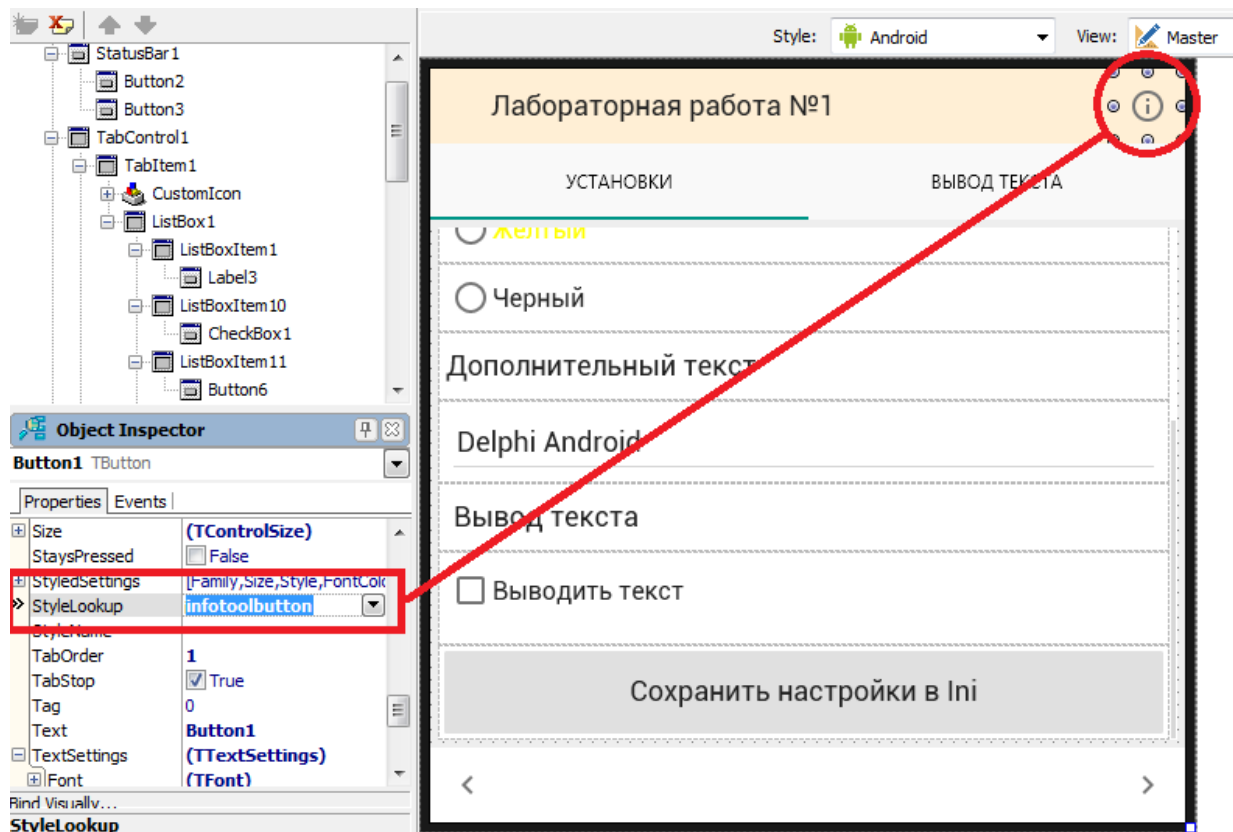


Рисунок 1.17 – Встановлення властивості StyleLookUp

Для введення додаткового тексту з палітри інструментів дістається однорядкове поле Edit і розташовується в ListBoxItem (рис. 1.18).



Рисунок 1.18 – Вкладення Edit у ListBoxItem

Можна змінювати властивість Text компонента Edit. Друга сторінка містить багаторядне поле редагування та кнопку для виведення тексту в це поле (рис. 1.19).

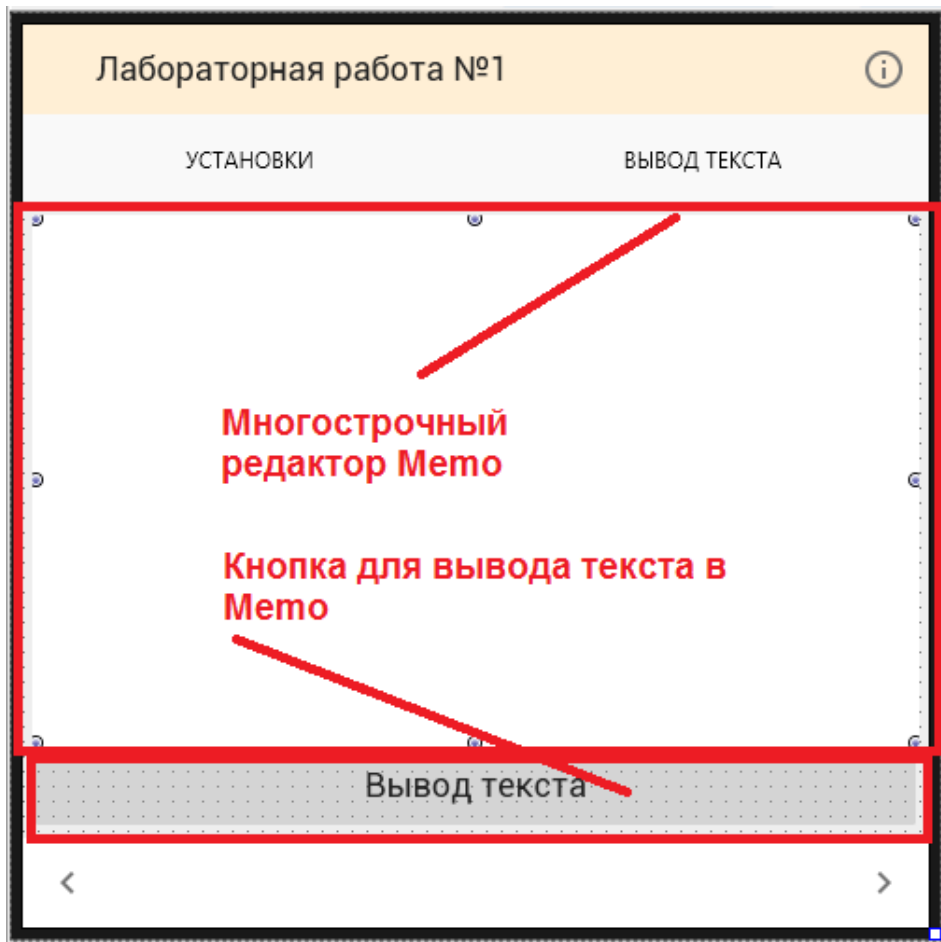


Рисунок 1.19 – Друга вкладка з багаторядним полем для виведення тексту

Для того, щоб компонент Мемо1 змінював налаштування шрифту програмно потрібно виконати наступне (рис. 1.20).

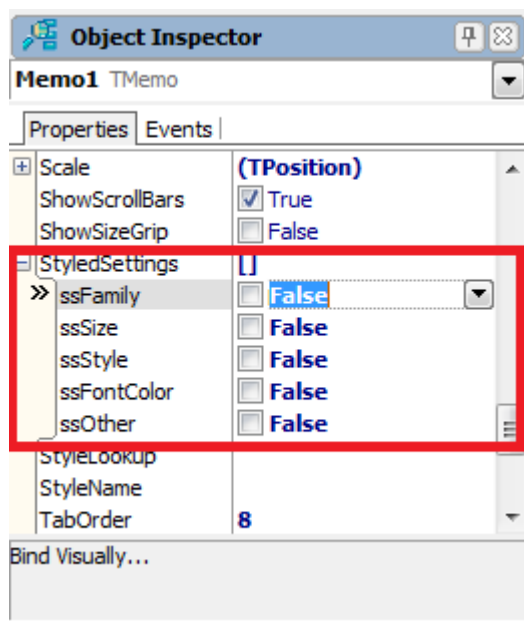



Рисунок 1.20 – Налаштування шрифту багаторядного поля

У даній лабораторній роботі будуть такі обробники подій: при натисканні на кнопку "Виведення тексту" (OnClick) і створенні головної форми (OnCreate), при натисканні на кнопці  для виведення другої форми з інформацією про програму, при натисканні на кнопки вперед і назад для перегортання сторінок TabControl, при показі головної форми (OnShow) для завантаження налаштувань і їх застосування з Ini файлу, при натисканні на кнопку "Зберегти налаштування в Ini".

При натисканні на кнопку "Виведення тексту" (рис. 1.21)

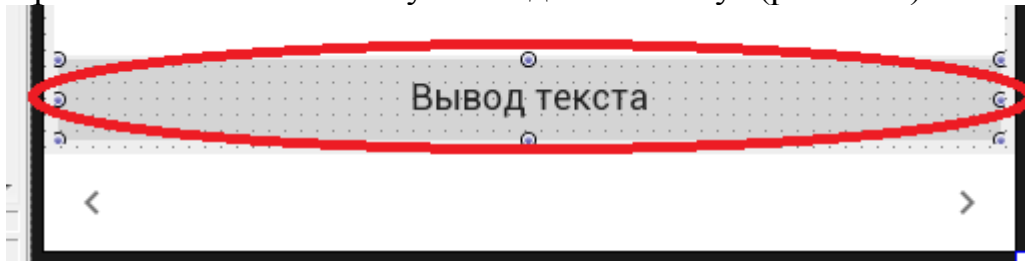


Рисунок 1.21 – Кнопка виводу тексту

потрібно написати наступний оброблювач:

```
procedure TMainForm.Button5Click (Sender: TObject);
var str: string; // змінна рядки, яка виводиться в поле
    fontsize: string; // змінна розміру шрифту
    textcolor: TAlphaColor; // Колір тексту
begin
    // Очищаємо поле
    Memo1.Lines.Clear;
    // Рядок для виведення
    str: = 'Hello World!';
    // Опитування перемикача Якщо встановлено - Висновок додаткового тексту
    if Checkbox1.IsChecked then
        begin
            str: = str + edit2.Text;
            Check: = 1;
        end
    else
        begin
            Check: = 0;
            str: = str;
        end;
    // текстова змінна для Ini файлу
    MyAddString: = Edit2.Text;
    // Введення розміру шрифту
    InputQuery ( 'Розмір шрифту', [ 'Введіть розмір шрифту'], [ '10'],
        procedure (const AResult: TModalResult; const AValues: array of string)
        begin
            case AResult of
                mrOk:
                    begin
```

```

fontsize: = AValues [0]; // розмір шрифту
if fontsize <> "" then // якщо не порожнє поле
    Memo1.Font.Size: = StrToInt (fontsize)
else
    Memo1.Font.Size: = 20;
// Встановлюємо колір шрифту
if Radiobutton1.IsChecked then
    begin
        textcolor: = TAlphaColors.Red;
        ColorSelected: = 1;
    end
else
    if Radiobutton2.IsChecked then
        begin
            textcolor: = TAlphaColors.Green;
            ColorSelected: = 2;
        end
    else
        if Radiobutton3.IsChecked then
            begin
                textcolor: = TAlphaColors.Blue;
                ColorSelected: = 3;
            end
        else
            if Radiobutton4.IsChecked then
                begin
                    textcolor: = TAlphaColors.Yellow;
                    ColorSelected: = 4;
                end
            else
                if Radiobutton5.IsChecked then
                    begin
                        textcolor: = TAlphaColors.Black;
                        ColorSelected: = 5;
                    end;
                Memo1.FontColor: = textcolor;
                // Виводимо рядок в текстове поле
                Memo1.Lines.Add (str);
            end;
        mrCancel:
        begin
        end;
    end;
end
);
end;

```


Для перемикання сторінок TabControl використовуються наступні обробники [9].

```
procedure TMainForm.Button2Click (Sender: TObject);
begin
    TabControl1.ActiveTab := TabItem2; // на вкладку 2
end;
```

```
procedure TMainForm.Button3Click (Sender: TObject);
begin
    TabControl1.ActiveTab := TabItem1; // на вкладку 1
end;
```

Для роботи з Ini файлами потрібно додати стандартну бібліотеку в uses (рис. 1.22).

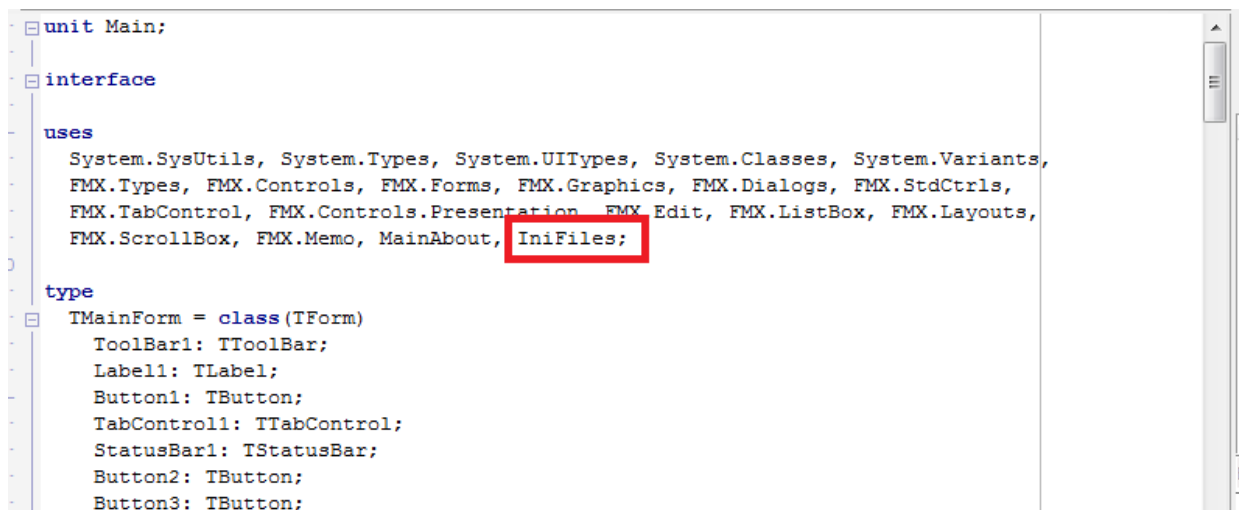


Рисунок 1.22 – Стандартна бібліотека для роботи з Ini – файлами

Оброблювач збереження налаштувань в Ini файл на SDCard мобільного пристрою:

```
procedure TMainForm.Button6Click (Sender: TObject);
var MyIni: TIniFile;
begin
    // Зберігаємо налаштування в Ini
    MyIni := TIniFile.Create ( '/mnt/sdcard/Lab1Conf.ini'); // Створення файлу
    MyIni.WriteString ( 'AddText', 'Value', MyAddString);
    MyIni.WriteInteger ( 'ColorSelect', 'Value', ColorSelected);
    MyIni.WriteInteger ( 'Check', 'Value', Check);
    MyIni.Free;
    ShowMessage ( 'Налаштування успішно збережені!');
end;
```

Увага! Для роботи з додатка з зовнішніми файлами потрібно встановити дозволи на читання і запис файлів (Меню Project> Options> Uses Permissions) (рис. 1.23 – 1.24)

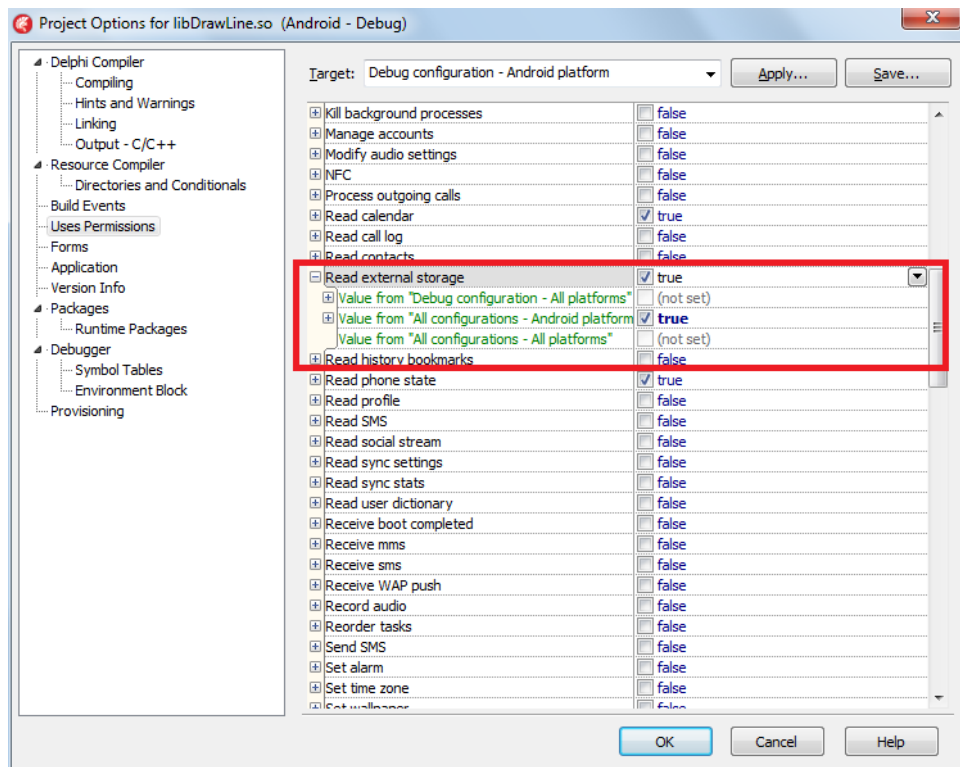


Рисунок 1.23 – Встановлення дозволів на зчитування карти пам'яті

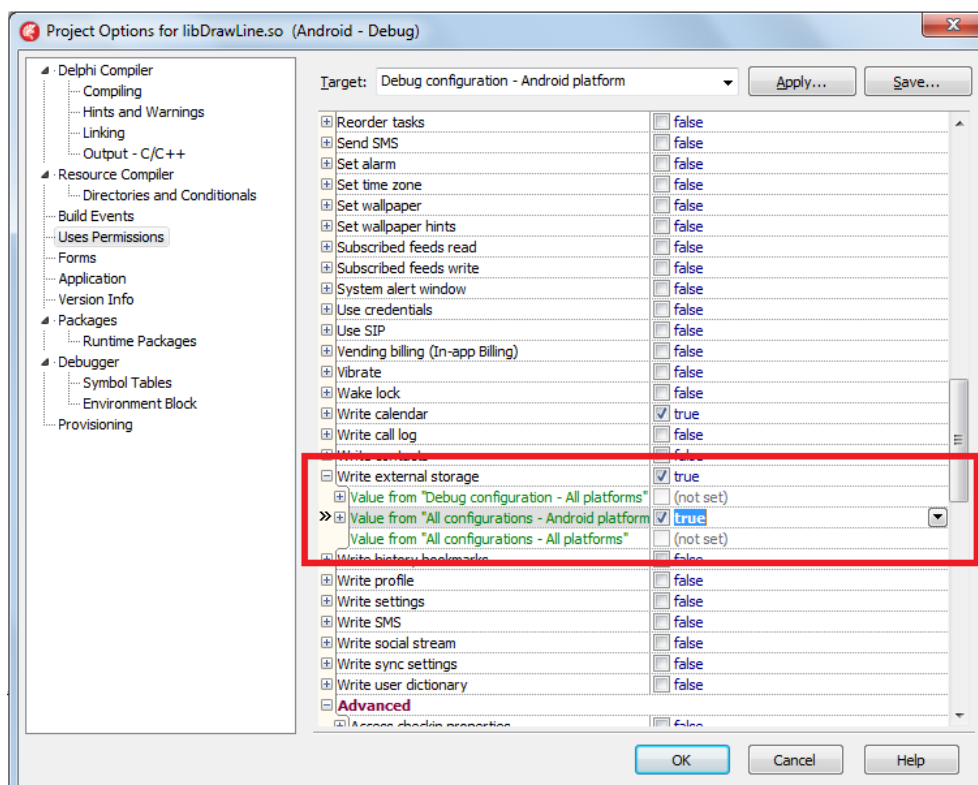


Рисунок 1.24 – Встановлення дозволів на запис карти пам'яті

Оброблювач при показі форми:

```
procedure TMainForm.FormShow (Sender: TObject);
var MyIni: TIniFile;
begin
  // Завантажуємо настройки з Ini
  if FileExists ( '/ mnt / sdcard / Lab1Conf.ini') then
  begin
    MyIni: = TIniFile.Create ( '/ mnt / sdcard / Lab1Conf.ini'); // Створення файлу
    MyAddString: = MyIni.ReadString ( 'AddText', 'Value', 'Delphi XE8 Droid');
    ColorSelected: = MyIni.ReadInteger ( 'ColorSelect', 'Value', 1);
    Check: = MyIni.ReadInteger ( 'Check', 'Value', 1); // 1 Секція, 2 значення, 3 за
замовчуванням
    // Встановлюємо налаштування інтерфейсу
    Edit2.Text: = MyAddString;

    if Check = 1 then
      CheckBox1.IsChecked: = true
    else
      CheckBox1.IsChecked: = false;
  // Встановлюємо перемикач
  case ColorSelected of
    1: RadioButton1.IsChecked: = true;
    2: RadioButton2.IsChecked: = true;
    3: RadioButton3.IsChecked: = true;
    4: RadioButton4.IsChecked: = true;
    5: RadioButton5.IsChecked: = true;
  end;
  MyIni.Free; // Прибираємо за собою
end;
end;
```

Оброблювач при створенні форми:

```
procedure TMainForm.FormCreate (Sender: TObject);
begin
  // Вибір кольору головної форми інтерфейсу
  MainForm.Fill.Kind: = TBrushKind.bkSolid; // Заливка форми суцільна
  MainForm.Fill.Color: = TAlphaColors.Aqua; // Установка кольору форми Aqua
end;
```

Для створення другої форми потрібно вибрати меню File-> New-> Multi-Device Form Delphi (рис. 1.25).

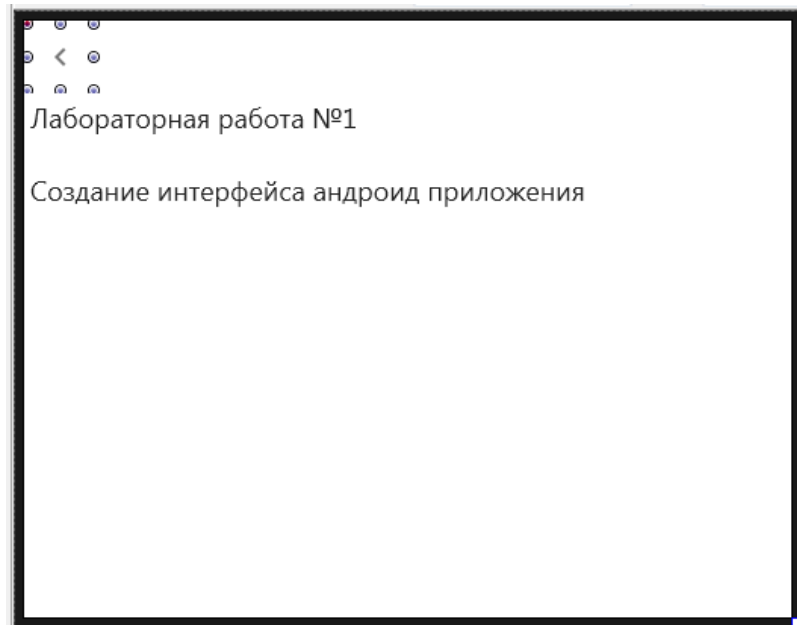


Рисунок 1.25 – Друга форма з інформацією

Інтерфейс форми з інформацією про програму розробляється аналогічно головній формі. Для зв'язку головної форми і підпорядкованої потрібно в розділ uses додати імена їх модулів (рис. 1.26).

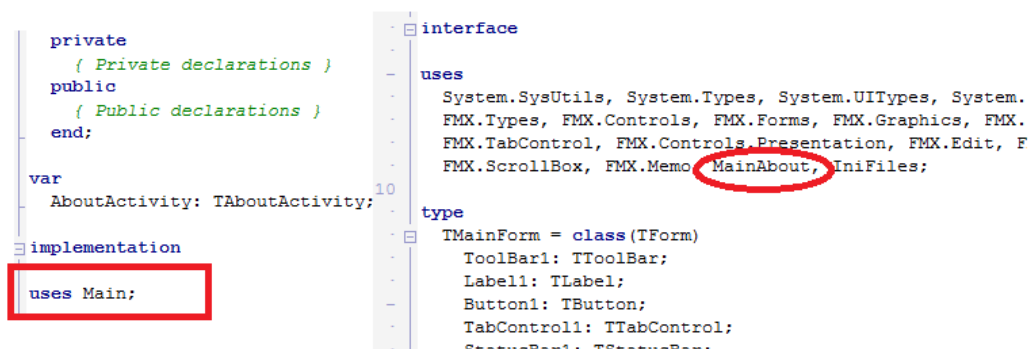


Рисунок 1.26 – Зв'язок головної та другої форми

Оброблювач клавіші повернення на головну форму

```

procedure TAboutActivity.Button1Click (Sender: TObject);
begin
  MainForm.Show; // Показ головною форми (повернення)
  AboutActivity.Visible := false; // приховування форми про програму
end;
  
```

Оброблювач клавіші показу форми про програму

```

// Показ форми про програму
procedure TMainForm.Button1Click (Sender: TObject);
begin
  AboutActivity.Show;
end;
  
```

Лістинг програми Main.pas

```
unit Main;
interface
uses
  System.SysUtils, System.Types, System.UITypes, System.Classes, System.Variants,
  FMX.Types, FMX.Controls, FMX.Forms, FMX.Graphics, FMX.Dialogs, FMX.StdCtrls,
  FMX.TabControl, FMX.Controls.Presentation, FMX.Edit, FMX.ListBox, FMX.Layouts,
  FMX.ScrollBox, FMX.Memo, MainAbout, IniFiles;
type
  TMainForm = class (TForm)
    ToolBar1: TToolBar;
    Label1: TLabel;
    Button1: TButton;
    TabControl1: TTabControl;
    StatusBar1: TStatusBar;
    Button2: TButton;
    Button3: TButton;
    TabItem1: TTabItem;
    TabItem2: TTabItem;
    ListBox1: TListBox;
    ListBoxItem1: TListBoxItem;
    ListBoxItem2: TListBoxItem;
    ListBoxItem3: TListBoxItem;
    ListBoxItem4: TListBoxItem;
    ListBoxItem5: TListBoxItem;
    ListBoxItem6: TListBoxItem;
    ListBoxItem7: TListBoxItem;
    ListBoxItem8: TListBoxItem;
    RadioButton1: TRadioButton;
    Label3: TLabel;
    RadioButton2: TRadioButton;
    RadioButton3: TRadioButton;
    RadioButton4: TRadioButton;
    RadioButton5: TRadioButton;
    ListBoxItem9: TListBoxItem;
    ListBoxItem10: TListBoxItem;
    Label4: TLabel;
    Edit2: TEdit;
    Label2: TLabel;
    Button4: TButton;
    Button5: TButton;
    Memo1: TMemo;
    CheckBox1: TCheckBox;
    ListBoxItem11: TListBoxItem;
    Button6: TButton;
    procedure Button3Click (Sender: TObject);
    procedure Button2Click (Sender: TObject);
    procedure Button5Click (Sender: TObject);
```

```

    procedure FormCreate (Sender: TObject);
    procedure Button1Click (Sender: TObject);
    procedure Button6Click (Sender: TObject);
    procedure FormShow (Sender: TObject);
private
    {Private declarations}
public
    {Public declarations}
end;
var
    MainForm: TMainForm;
    ColorSelected: integer; // обраний колір
    MyAddString: string; // додатковий рядок
    Check: integer; // змінна для зберігання значення, який перемикач натиснуто

```

Implementation

```

{$ R * .fmx}
// Показ форми про програму
procedure TMainForm.Button1Click (Sender: TObject);
begin
    AboutActivity.Show;
end;
procedure TMainForm.Button2Click (Sender: TObject);
begin
    TabControl1.ActiveTab: = TabItem2;
end;

procedure TMainForm.Button3Click (Sender: TObject);
begin
    TabControl1.ActiveTab: = TabItem1;
end;

procedure TMainForm.Button5Click (Sender: TObject);
var str: string; // змінна рядки, яка виводиться в поле
    fontsize: string; // змінна розміру шрифту
    textcolor: TAlphaColor; // Колір тексту
begin
    // Очищаємо поле
    Memo1.Lines.Clear;
    // Рядок для виведення
    str: = 'Hello World!';
    // Опитування перемикача Якщо встановлено - Висновок додаткового тексту
    if Checkbox1.IsChecked then
    begin
        str: = str + edit2.Text;
        Check: = 1;
    end
    else

```

```

begin
    Check: = 0;
    str: = str;
end;
// текстова змінна для Ini файлу
MyAddString: = Edit2.Text;
// Введення розміру шрифту
InputQuery ( 'Розмір шрифту', [ 'Введіть розмір шрифту'], [ '10'],
    procedure (const AResult: TModalResult; const AValues: array of string)
    begin
        case AResult of
            mrOk:
                begin
                    fontsize: = AValues [0];
                    if fontsize <> '' then
                        Memo1.Font.Size: = StrToInt (fontsize)
                    else
                        Memo1.Font.Size: = 20;

                    // Встановлюємо колір шрифту
                    if Radiobutton1.IsChecked then
                        begin
                            textcolor: = TAlphaColors.Red;
                            ColorSelected: = 1;
                        end
                    else
                        if Radiobutton2.IsChecked then
                            begin
                                textcolor: = TAlphaColors.Green;
                                ColorSelected: = 2;
                            end
                        else
                            if Radiobutton3.IsChecked then
                                begin
                                    textcolor: = TAlphaColors.Blue;
                                    ColorSelected: = 3;
                                end
                            else
                                if Radiobutton4.IsChecked then
                                    begin
                                        textcolor: = TAlphaColors.Yellow;
                                        ColorSelected: = 4;
                                    end
                                else
                                    if Radiobutton5.IsChecked then
                                        begin
                                            textcolor: = TAlphaColors.Black;
                                            ColorSelected: = 5;
                                        end
                                    end;
                                end;
                end
        end
    end;

```

```

        Memo1.FontColor := textcolor;
        // Виводимо рядок в текстове поле
        Memo1.Lines.Add (str);
    end;
    mrCancel:
    begin
    end;
end;
end;
);
end;
procedure TMainForm.Button6Click (Sender: TObject);
var MyIni: TIniFile;
begin
    // Зберігаємо налаштування в Ini
    MyIni := TIniFile.Create ( '/ mnt / sdcard / Lab1Conf.ini'); // Створення файлу
    MyIni.WriteString ( 'AddText', 'Value', MyAddString);
    MyIni.WriteInteger ( 'ColorSelect', 'Value', ColorSelected);
    MyIni.WriteInteger ( 'Check', 'Value', Check);
    MyIni.Free;
    Showmessage ( 'Налаштування успішно збережені!');
end;
procedure TMainForm.FormCreate (Sender: TObject);
begin
    // Вибір кольору головної форми інтерфейсу
    MainForm.Fill.Kind := TBrushKind.bkSolid; // Заливка форми суцільна
    MainForm.Fill.Color := TAlphaColors.Aqua; // Установка кольору форми Aqua
end;
procedure TMainForm.FormShow (Sender: TObject);
var MyIni: TIniFile;
begin
    // Завантажуємо настройки з Ini
    if FileExists ( '/ mnt / sdcard / Lab1Conf.ini') then
    begin
        MyIni := TIniFile.Create ( '/ mnt / sdcard / Lab1Conf.ini'); // Створення файлу
        MyAddString := MyIni.ReadString ( 'AddText', 'Value', 'Delphi XE8 Droid'); // 1 Секція,
        2 значення, 3 за замовчуванням
        ColorSelected := MyIni.ReadInteger ( 'ColorSelect', 'Value', 1);
        Check := MyIni.ReadInteger ( 'Check', 'Value', 1);
        // Встановлюємо налаштування інтерфейсу
        Edit2.Text := MyAddString;
        if Check = 1 then
            CheckBox1.IsChecked := true
        else
            CheckBox1.IsChecked := false;
        case ColorSelected of
            1: RadioButton1.IsChecked := true;
            2: RadioButton2.IsChecked := true;
            3: RadioButton3.IsChecked := true;

```



```

4: RadioButton4.IsChecked: = true;
5: RadioButton5.IsChecked: = true;
end;
MyIni.Free;
end;
end;
end.

```

Лістинг MainAbout.pas

```

unit MainAbout;
interface
uses
  System.SysUtils, System.Types, System.UITypes, System.Classes, System.Variants,
  FMX.Types, FMX.Controls, FMX.Forms, FMX.Graphics, FMX.Dialogs, FMX.StdCtrls,
  FMX.Controls.Presentation, FMX.ScrollBox, FMX.Memo;
type
  TAboutActivity = class (TForm)
    ToolBar1: TToolBar;
    Button1: TButton;
    Memo1: TMemo;
    procedure Button1Click (Sender: TObject);
  private
    {Private declarations}
  public
    {Public declarations}
  end;
var
  AboutActivity: TAboutActivity;
implementation
uses Main;
{$R *.fmx}
procedure TAboutActivity.Button1Click (Sender: TObject);
begin
  MainForm.Show;
  AboutActivity.Visible: = false;
end;
end.

```

Зберегти проект і модулі проекту можна за допомогою меню File> Save Project as (Save as). *Потрібно зберігати в одну папку.*

Коли розробка проекту завершена, його потрібно скопіювати і побудувати (при цьому створюється файл .apk, який можна встановлювати на андроїд пристрої, проте в налаштуваннях пристрою потрібно дозволити установку з невідомих джерел, тобто не тільки з PlayMarket). Для налагодження програми можна використовувати або віртуальний емулятор, або реальне андроїд пристрій. Для налагодження і запуску додатка на реальному пристрої потрібно встановити Google USB Driver. Всі скріншоти наведені на рис. 1.27.

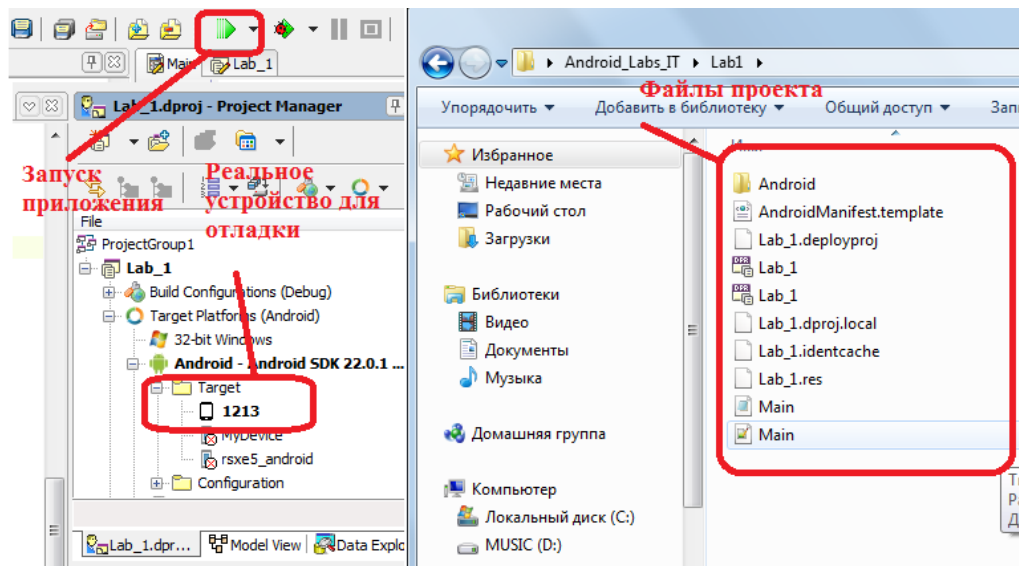


Рисунок 1.27 – Налаштовування додатку на реальному мобільному пристрої

Розташування скомпільованої арк файлу андроїд додатка в папках створеного проекту (рис. 1.28).

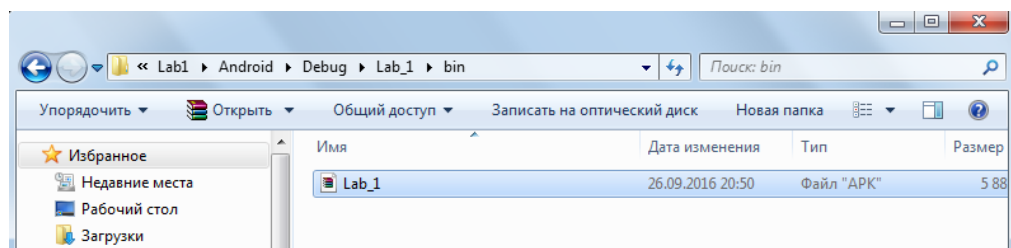


Рисунок 1.28 – Скомпільований арк архів

Даний арк файл може бути встановлений на андроїд пристрої.

Індивідуальне завдання до лабораторної роботи

1. Для всіх об'єктів проекту змінити властивість Name у інспекторі об'єктів за зразком: ПІБ_Оригінальне ім'я. Наприклад Babash_AV_Button1, Babash_AV_Memo1 (латиниця) (рис. 1.29).

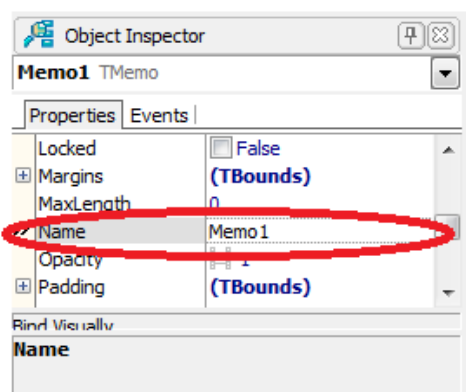


Рисунок 1.29 – Встановлення властивості Name

2. Зібрати арк файл і встановити на реальний андроїд пристрій для демонстрації працездатності.

Зауваження. Для нормальної компіляції і збірки арк файлу необхідно вибрати іконку png для вашого мобільного додатка (Меню Project> Options> Application). Якщо іконка не вибрана, мобільне застосування не скомпілюється і не створиться арк (рис. 1.30).

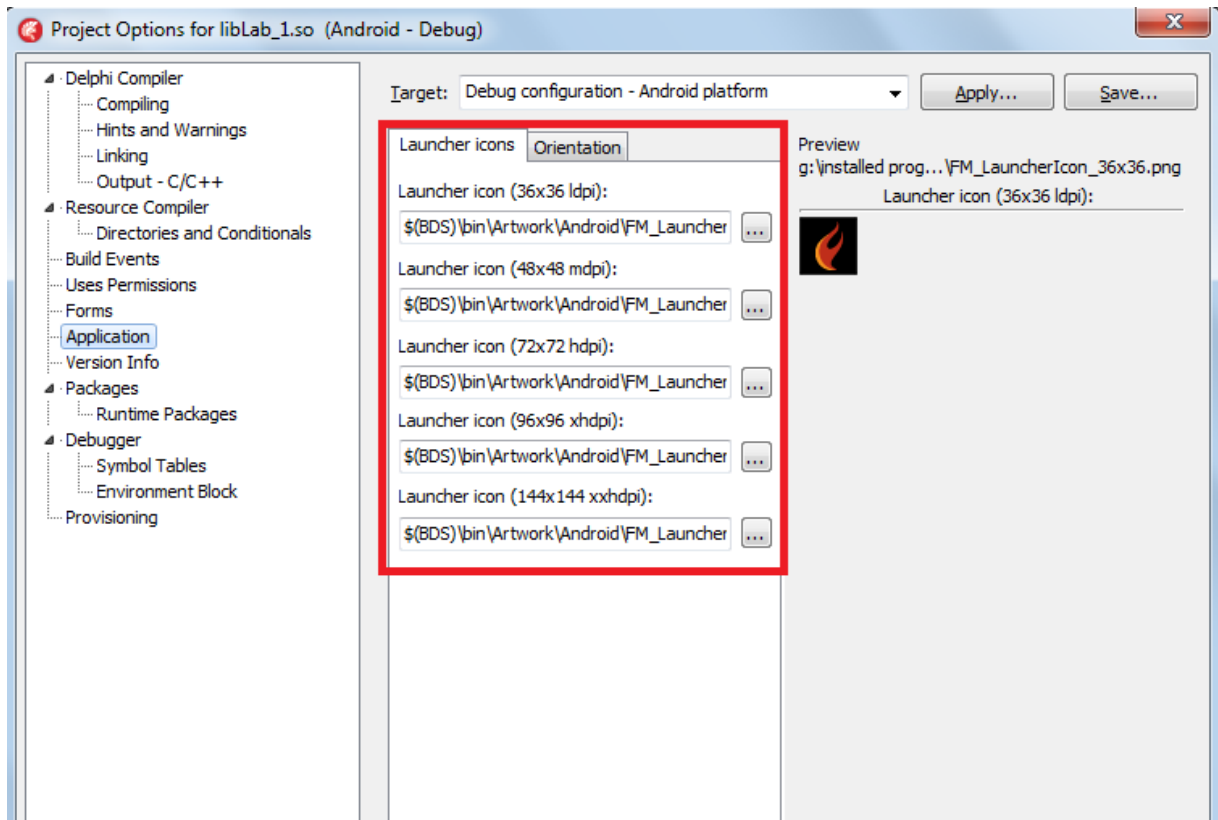


Рисунок 1.30 – Вибір іконки для андроїд – додатку

Контрольні запитання

1. Що таке Ini файли? Для чого використовуються?
2. Поясніть, для чого використовується властивість будь-якого об'єкта Name?
3. Поясніть різницю між TRadioButton та TCheckBox?

2 ЛАБОРАТОРНА РОБОТА №2

Тема. Робота з графікою. Обробка натискань функціональних клавіш. Використання і обробка жестів.

Мета. Освоєння роботи з графікою FireMonkey, навчитися програмно малювати використовуючи Canvas. Навчитися обробляти натискання різних функціональних клавіш мобільного пристрою. Навчитися використовувати різні жести в мобільних додатках.

Хід роботи

Доступ до графічної поверхні головної форми здійснюється за допомогою об'єкта TCanvas. Графічна поверхня являє собою сукупність пікселів. Положення пікселя по горизонтальній осі визначається координатою X і по вертикальній осі координатою Y.

Принцип побудови графіки за допомогою Canvas полягає у виведенні зображення на екран при настанні події Paint, тобто при перемальовуванні канви. Таким чином, система координат екрану мобільного пристрою має вигляд рис. 2.1.

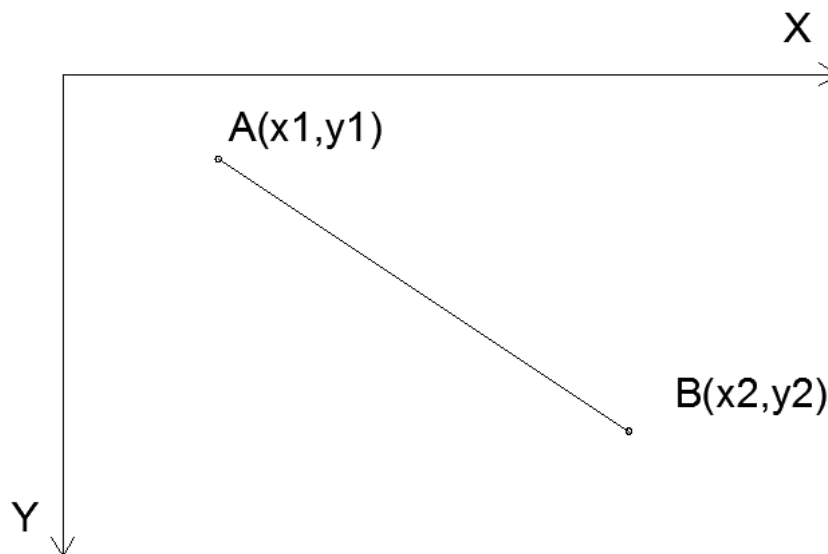


Рисунок 2.1 – Система координат екрану мобільного пристрою

Мова Delphi має клас TPoint (Точка). Об'єкт класу TPoint має координати X і Y. Наприклад точка A має координати X1, Y1.

Приклад ініціалізації точки в Delphi.

```
var p1: TPoint;
```

```
.....
```

```
p1.X: = 100;
```

```
p1.Y: = 200;
```

Для виведення ліній в Canvas має метод DrawLine. Приклад конкретного використання:

```
Form1.Canvas.DrawLine (p1, p2,1.0);
```

Метод DrawLine малює лінію поєднуючи точки p1, p2. AOpacity - ступінь прозорості лінії (0..1.5).

Для встановлення кольору лінії, типу лінії (суцільна, штрих-пунктирна, точкова) товщини лінії використовуються відповідно наступні команди:

```
Form1.Canvas.Stroke.Color:= TAlphaColors.Aqua; // колір Аква
```

```
Form1.Canvas.Stroke.Dash:= TStrokeDash.Dot; // Лінія точками
```

```
Form1.Canvas.Stroke.Thickness:= 2; // Товщина лінії
```

При створенні проекту потрібно встановити колір форми відповідним чином (рис. 2.2).

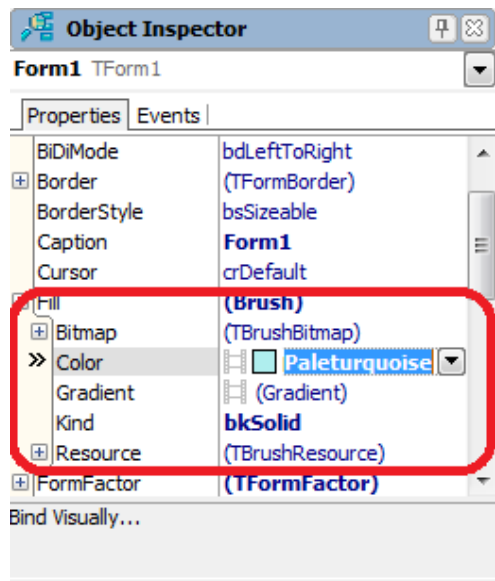


Рисунок 2.2 – Встановлення заливки форми

Малювати графіку слід в блоках BeginScene і EndScene (спочатку відкрити сцену а потім закрити).

Для малювання на формі будь-якої графіки потрібно створити обробник події головної форми OnPaint.

У Delphi для малювання графіки є спеціальний компонент PaintBox, який і буде використаний для малювання в даній лабораторній роботі. Даний компонент також як і Form має властивість Canvas.

Робота з жестами за допомогою GestureManager

В сучасних мобільних додатках дуже часто зустрічається використання різних жестів. Наприклад, при перегляді зображень можливо їх збільшення / зменшення за допомогою щипків пальцями, або її поворот при поверненні двома пальцями. Також можна проводити перегортання сторінок електронної книги при проведенні пальцем по екрану вліво-вправо.

У Delphi є спеціальний компонент для роботи з жестами TGestureManager (рис. 2.3).

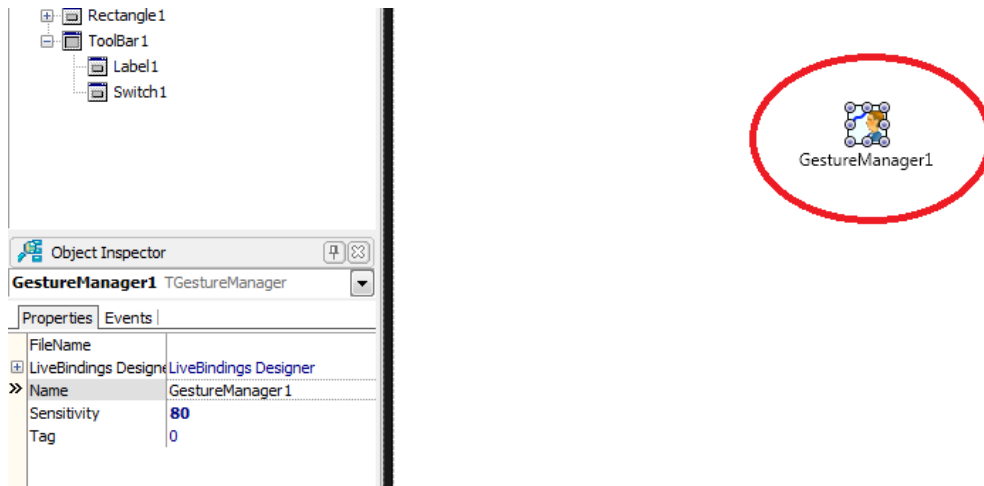


Рисунок 2.3 – Встановлення заливки форми

Обробляти жести можна за допомогою вистеження події OnGesture, наприклад Form1Gesture.

Однак для того, щоб було можливо обробляти різні жести необхідно виконати певні налаштування, тобто дозволити використання певних жестів (даному випадку в головній формі Form1). Можна використовувати стандартні та інтерактивні жести. У даній лабораторній роботі будуть використані стандартні жести (рис. 2.4).

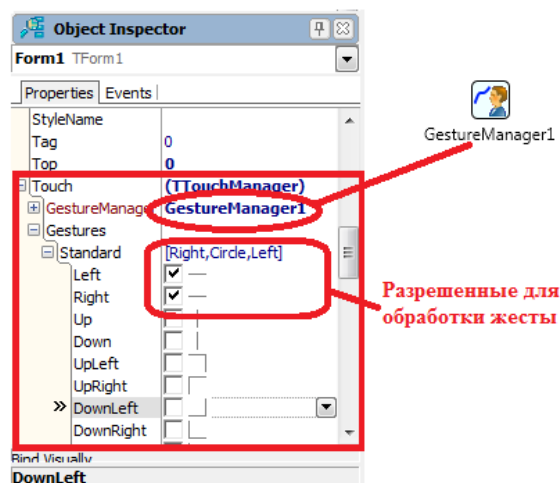


Рисунок 2.4 – Налаштування потрібних жестів

Таким чином, в даному випадку дозволені такі жести: проведення пальцем вліво по формі (Left), проведення пальцем вправо (Right), жест у вигляді описування пальцем кола по екрану (Circle).

Приклад обробки жестів наведено нижче.

```
procedure TForm1.FormGesture (Sender: TObject;
  const EventInfo: TGestureEventInfo; var Handled: Boolean);
begin
  // Визначаємо доступний жест і обробляємо його
  case EventInfo.GestureID of
    // Рух вліво (збільшуємо товщину лінії виведеної графіки)
```

```

sgLeft:
begin
    ThicknessInc (Label2);
    PaintBox1.Repaint;
end;
// Рух вправо (зменшуємо товщину лінії виведеної графіки)
sgRight:
begin
    ThicknessDec (Label2);
    PaintBox1.Repaint;
end;
// При промальовуванні кола
// Показуємо інтерфейс настройки типу лінії і її кольору
sgCircle:
begin
    // показуємо настройки
    Rectangle1.Visible: = true;
    // перемальовували PaintBox (він служить для виведення графіки)
    PaintBox1.Repaint;
end;
end;
end;

```

В даному випадку ми обробляємо подія OnGesture головної форми. При русі пальцем вліво збільшуємо товщину ліній виведеної графіки і перемальовували PaintBox. При русі вправо - зменшуємо товщину ліній.

При описуванні кола на формі виводимо налаштування ліній графіки (Компонент Rectangle) (рис. 2.5).

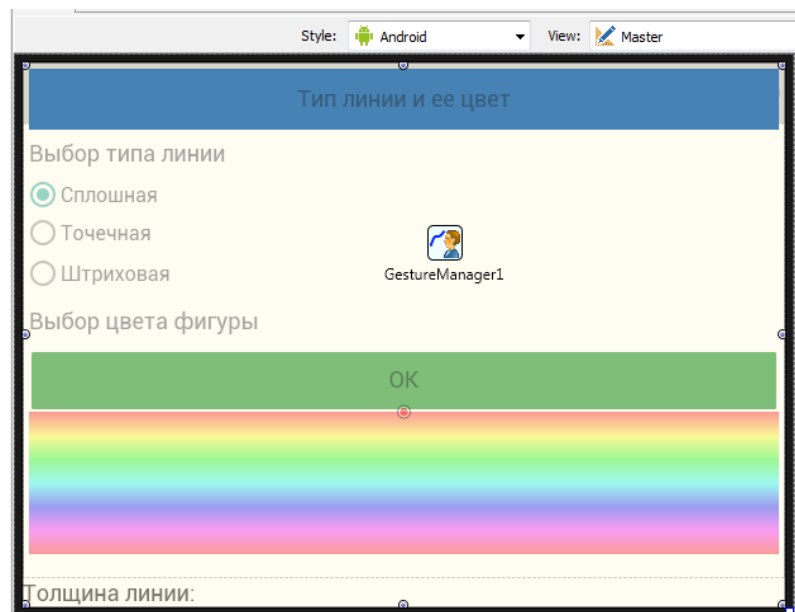


Рисунок 2.5 – Інтерфейс налаштувань типу та кольору ліній

Для вибору кольору лінії використовується компонент ColorPicker.

Зауваження. Властивість *Opacity* дозволяє задати ступінь прозорості будь-якого компонента інтерфейсу (значення 0..1.0). Може бути корисним при створенні різних красивих ефектів при проектуванні складних інтерфейсів (рис. 2.6).

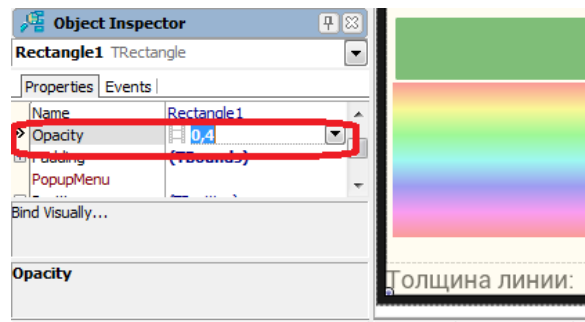


Рисунок 2.6 – Властивість *Opacity*

Компонент *Rectangle* містить в собі безліч дочірніх компонентів для формування інтерфейсу налаштування відображення ліній на канві (*Canvas*). Його як будь-який інший можна приховувати під час проектування, щоб він не заважав працювати.

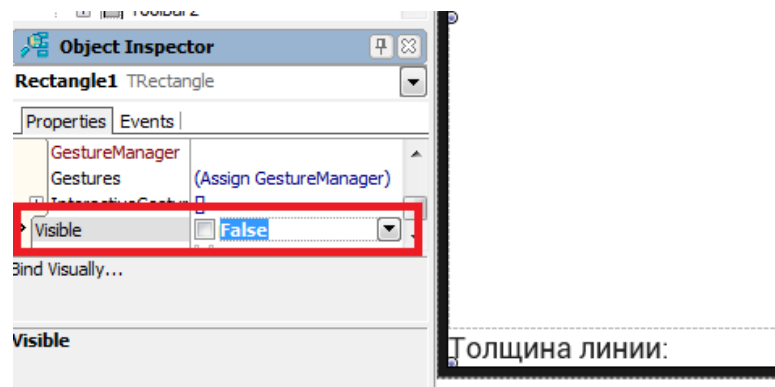


Рисунок 2.7 – Встановлення заливки форми

Власне малювання на канві *Canvas PaintBox* відбуватиметься за подією *OnPaint*.

```
procedure TForm1.PaintBox1Paint (Sender: TObject; Canvas: TCanvas);
var p1, p2: tpoint; // точки p1 і p2
    x0, y0: integer; // координати центру
begin
    // Колір форми
    PaintBox1.Canvas.Stroke.Color := ColorPicker1.Color;
    // Вибір типу лінії
    case Itsel of
        1: PaintBox1.Canvas.Stroke.Dash := TStrokeDash.Solid; // Сплошна
        2: PaintBox1.Canvas.Stroke.Dash := TStrokeDash.Dot; // Точкова
        3: PaintBox1.Canvas.Stroke.Dash := TStrokeDash.Dash; // Штрихова
    end;
end;
```



```

    PaintBox1.Canvas.Stroke.Thickness: = thickness; // Товщина лінії
// Малювати графіку слід між блоками BeginScene і EndScene
    PaintBox1.Canvas.BeginScene;
// Координати центру форми (Round округлює число до цілого)
    x0: = Round (PaintBox1.Width / 2);
    y0: = Round (PaintBox1.Height / 2);
// Перша лінія
    p1.X: = x0 + size;
    p1.Y: = y0 + size;
    p2.X: = x0 + size;
    p2.Y: = y0-size;
    PaintBox1.Canvas.DrawLine (p1, p2,1.0);
// Друга лінія
    p1.X: = x0 + size;
    p1.Y: = y0 + size;
    p2.X: = x0-size;
    p2.Y: = y0 + size;
    PaintBox1.Canvas.DrawLine (p1, p2,1.0);
// Третя лінія
    p1.X: = x0-size;
    p1.Y: = y0 + size;
    p2.X: = x0-size;
    p2.Y: = y0-size;
    PaintBox1.Canvas.DrawLine (p1, p2,1.0);
// Четверта лінія
    p1.X: = x0-size;
    p1.Y: = y0-size;
    p2.X: = x0 + size;
    p2.Y: = y0-size;
    PaintBox1.Canvas.DrawLine (p1, p2,1.0);
// Перехресне 1
    p1.X: = x0-size;
    p1.Y: = y0-size;
    p2.X: = x0 + size;
    p2.Y: = y0 + size;
    PaintBox1.Canvas.DrawLine (p1, p2,1.0);
// Перехресне 2
    p1.X: = x0-size;
    p1.Y: = y0 + size;
    p2.X: = x0 + size;
    p2.Y: = y0-size;
    PaintBox1.Canvas.DrawLine (p1, p2,1.0);
    PaintBox1.Canvas.EndScene;
end;

```

Робота з програмно-апаратними клавішами андроїд пристрої

Часто при розробці програми під мобільні пристрої потрібно обробляти натискання на хардварний кнопки (наприклад "Меню / Назад" і т.п.). Для

обробки натискань клавіш необхідно відстежувати події OnKeyUp (відпускання) і OnKeyDown (натискання) головною форми.

Список кодів для часто використовуваних кнопок:

Назад - 137 - vkHardwareBack

Гучність збільшити - 175 - \$ AF - vkVolumeUp

Гучність зменшити - 174 - \$ AE - vkVolumeDown

Наведемо приклад обробки натискання клавіш гучності pf хардварної кнопки Назад. У даній лабораторній роботі буде змінюватися розмір відображуваного квадрата з діагоналями на канві PaintBox при натисканні клавіш гучності.

```
// Натискання клавіші
```

```
procedure TForm1.FormKeyDown (Sender: TObject; var Key: Word; var KeyChar: Char;  
Shift: TShiftState);
```

```
begin
```

```
// Кнопка гучності вгору
```

```
if Key = vkVolumeUp then
```

```
begin
```

```
    Key: = 0;
```

```
    size: = size-5;
```

```
if size <5 then size: = 5;
```

```
    PaintBox1.Repaint;
```

```
end;
```

```
// Кнопка гучності вниз
```

```
if Key = vkVolumeDown then
```

```
begin
```

```
    Key: = 0;
```

```
    size: = size + 5;
```

```
if size > 150 then size: = 150;
```

```
    PaintBox1.Repaint;
```

```
end;
```

```
// При натисканні тому підтвердження виходу з програми
```

```
if Key = vkHardwareBack then
```

```
begin
```

```
    Key: = 0;
```

```
    ExitConfirm;
```

```
end;
```

```
end;
```

При відпусканні клавіш просто блокуємо їх системні функції

```
// Відпускання кнопки
```

```
procedure TForm1.FormKeyUp (Sender: TObject; var Key: Word; var KeyChar: Char;  
Shift: TShiftState);
```

```
Begin
```

```

if Key = vkVolumeUp then
begin
    Key: = 0; // Нічого не робимо
end;

```

```

if Key = vkVolumeDown then
begin
    Key: = 0;
end;

```

```

if Key = vkHardwareBack then
begin
    Key: = 0;
end;
end;

```

Замість "Key = vkHardwareBack", можна писати "Key = 137". Це все, що потрібно для обробки натискання на кнопки.

Лістинг програми

```

unit Main;
interface
uses
    System.SysUtils, System.Types, System.UITypes, System.Classes, System.Variants,
    FMX.Types, FMX.Controls, FMX.Forms, FMX.Graphics, FMX.Dialogs, FMX.StdCtrls,
    FMX.Controls.Presentation, FMX.Colors, FMX.Gestures, FMX.Objects,
    FMX.Platform.Android;
type
    TForm1 = class (TForm)
        ToolBar1: TToolBar;
        Label1: TLabel;
        ColorPicker1: TColorPicker;
        Label2: TLabel;
        GestureManager1: TGestureManager;
        Rectangle1: TRectangle;
        RadioButton1: TRadioButton;
        Label3: TLabel;
        RadioButton2: TRadioButton;
        RadioButton3: TRadioButton;
        Button2: TButton;
        PaintBox1: TPaintBox;
        ToolBar2: TToolBar;
        Label4: TLabel;
        Label5: TLabel;
        Switch1: TSwitch;
        procedure FormKeyDown (Sender: TObject; var Key: Word; var KeyChar: Char;
            Shift: TShiftState);
        procedure FormKeyUp (Sender: TObject; var Key: Word; var KeyChar: Char;

```

```

    Shift: TShiftState);
procedure FormCreate (Sender: TObject);
procedure FormGesture (Sender: TObject; const EventInfo: TGestureEventInfo;
    var Handled: Boolean);
procedure Button1Click (Sender: TObject);
procedure Button2Click (Sender: TObject);
procedure PaintBox1Paint (Sender: TObject; Canvas: TCanvas);
procedure Switch1Switch (Sender: TObject);
private
    {Private declarations}
public
    {Public declarations}
end;
var
    Form1: TForm1;
    thickness: integer;
    ltsel: integer;
    size: integer;
implementation
{$ R * .fmx}
{$ R * .LgXhdpiPh.fmx ANDROID}
{$ R * .GGlass.fmx ANDROID}
{$ R * .Macintosh.fmx MACOS}
{$ R * .Windows.fmx MSWINDOWS}
// Підтвердження виходу з програми
procedure ExitConfirm;
begin
    // Висновок діалогового вікна
    MessageDlg ( 'Хочете вийти!', System.UITypes.TMsgDlgType.mtInformation,
    [
        System.UITypes.TMsgDlgBtn.mbYes,
        System.UITypes.TMsgDlgBtn.mbNo
    ], 0,
    procedure (const AResult: TModalResult)
    begin
        case AResult
        of
            mrYES:
                begin
                    MainActivity.finish; // Вихід з програми
                end;
            mrNo:
                begin
                    end;
                end;
        end;
    end
    )
end;
end;

```

```

// Зміна товщини лінії
procedure ThicknessInc (var l: TLabel);
begin
    thickness: = thickness + 1;
    if thickness > 10 then thickness: = 10;
    l.Text: = 'Товщина лінії:' + inttostr (thickness);
end;
procedure ThicknessDec (var l: TLabel);
begin
    thickness: = thickness-1;
    if thickness <1 then thickness: = 1;
    l.Text: = 'Товщина лінії:' + inttostr (thickness)
end;
// Вибір типу лінії
procedure TForm1.Button2Click (Sender: TObject);
begin
    if radiobutton1.IsChecked then ltsel: = 1;
    if radiobutton2.IsChecked then ltsel: = 2;
    if radiobutton3.IsChecked then ltsel: = 3;
    Rectangle1.Visible: = false;
end;
procedure TForm1.FormCreate (Sender: TObject);
begin
    thickness: = 1;
    size: = 40;
    ltsel: = 1;
    Label2.Text: = 'Товщина лінії:' + inttostr (thickness);
end;
procedure TForm1.FormGesture (Sender: TObject;
    const EventInfo: TGestureEventInfo; var Handled: Boolean);
begin
    // Визначаємо доступний жест і обробляємо його
    case EventInfo.GestureID of
    // Рух вліво (збільшуємо товщину лінії виведеної графіки)
    sgiLeft:
        begin
            ThicknessInc (Label2);
            PaintBox1.Repaint;
        end;
    // Рух вправо (зменшуємо товщину лінії виведеної графіки)
    sgiRight:
        begin
            ThicknessDec (Label2);
            PaintBox1.Repaint;
        end;
    // При промальовуванні кола
    // Показуємо інтерфейс налаштування типу лінії і її кольору
    sgiCircle:

```

```

begin
// показуємо настройки
Rectangle1.Visible: = true;
// перемальовували PaintBox (він служить для виведення графіки)
PaintBox1.Repaint;
end;
end;
end;
// Натискання клавіші
procedure TForm1.FormKeyDown (Sender: TObject; var Key: Word; var KeyChar: Char;
Shift: TShiftState);
begin
if Key = vkVolumeUp then
begin
Key: = 0;
size: = size-5;
if size <5 then size: = 5;
PaintBox1.Repaint;
end;
if Key = vkVolumeDown then
begin
Key: = 0;
size: = size + 5;
if size > 150 then size: = 150;
PaintBox1.Repaint;
end;
if Key = vkHardwareBack then
begin
Key: = 0;
ExitConfirm;
end;
end;
end;
// Відпускання кнопки
procedure TForm1.FormKeyUp (Sender: TObject; var Key: Word; var KeyChar: Char;
Shift: TShiftState);
begin
if Key = vkVolumeUp then
begin
Key: = 0;
end;
if Key = vkVolumeDown then
begin
Key: = 0;
end;
if Key = vkHardwareBack then
begin
Key: = 0;
end;
end;

```

```

end;
procedure TForm1.PaintBox1Paint (Sender: TObject; Canvas: TCanvas);
  var p1, p2: tpoint; // точки p1 і p2
  x0, y0: integer; // координати центру
begin
  // Колір форми
  PaintBox1.Canvas.Stroke.Color: = ColorPicker1.Color;
  // Вибір типу лінії
  case Itsel of
    1: PaintBox1.Canvas.Stroke.Dash: = TStrokeDash.Solid; // Сплошна
    2: PaintBox1.Canvas.Stroke.Dash: = TStrokeDash.Dot; // Точкова
    3: PaintBox1.Canvas.Stroke.Dash: = TStrokeDash.Dash; // Штрихова
  end;
  PaintBox1.Canvas.Stroke.Thickness: = thickness; // Товщина лінії
  // Малювати графіку слід між блоками BeginScene і EndScene
  PaintBox1.Canvas.BeginScene;
  // Координати центру форми (div розподіл без остачі)
  x0: = Round (PaintBox1.Width / 2);
  y0: = Round (PaintBox1.Height / 2);
  // Перша лінія
  p1.X: = x0 + size;
  p1.Y: = y0 + size;
  p2.X: = x0 + size;
  p2.Y: = y0-size;
  PaintBox1.Canvas.DrawLine (p1, p2,1.0);
  // Друга лінія
  p1.X: = x0 + size;
  p1.Y: = y0 + size;
  p2.X: = x0-size;
  p2.Y: = y0 + size;
  PaintBox1.Canvas.DrawLine (p1, p2,1.0);
  // Третя лінія
  p1.X: = x0-size;
  p1.Y: = y0 + size;
  p2.X: = x0-size;
  p2.Y: = y0-size;
  PaintBox1.Canvas.DrawLine (p1, p2,1.0);
  // Четверта лінія
  p1.X: = x0-size;
  p1.Y: = y0-size;
  p2.X: = x0 + size;
  p2.Y: = y0-size;
  PaintBox1.Canvas.DrawLine (p1, p2,1.0);
  // Перехресне 1
  p1.X: = x0-size;
  p1.Y: = y0-size;
  p2.X: = x0 + size;
  p2.Y: = y0 + size;

```

```

PaintBox1.Canvas.DrawLine (p1, p2,1.0);
// Перехресне 2
p1.X:= x0-size;
p1.Y:= y0 + size;
p2.X:= x0 + size;
p2.Y:= y0-size;
PaintBox1.Canvas.DrawLine (p1, p2,1.0);
PaintBox1.Canvas.EndScene;
end;
// Підтвердження виходу з програми при перемиканні Switch
procedure TForm1.Switch1Switch (Sender: TObject);
begin
if Switch1.IsChecked then
begin
ExitConfirm;
Switch1.IsChecked:= false;
end;
end;
end.

```

Для виходу з програми використовується перемикач Switch. Використовується його подія OnSwitch. При його перемиканні викликається обробник події перемикачання. Зовнішній вигляд приведений на рис. 2.8.



Рисунок 2.8 – Зовнішній вигляд перемикача

Індивідуальне завдання

Скомпілювати наведений приклад, встановити додаток на мобільний пристрій. Перевірити функціонування програми та проаналізувати результат роботи.

Вивести на екран мобільного пристрою зображення за допомогою властивості Canvas компонента PaintBox. Товщину лінії змінювати при натисканні клавіш гучність (VolumeUp, VolumeDown). За допомогою кругового жесту завершувати роботу мобільного додатка. Здійснити вибір типу ліній за допомогою TRadioButton. Варіанти індивідуальних завдань наведені у табл. 2.1.

Таблиця 2.1 – Варіанти індивідуальних завдань

№ Варіанта	Індивідуальне завдання
1	Вивести на екран букву "А". Колір червоний.
2	Вивести на екран букву "С". Колір зелений.
3	Вивести на екран букву "Е". Колір чорний.
4	Вивести на екран трикутник. Колір жовтий.
5	Вивести на екран прямокутник. Колір чорний.
6	Вивести на екран букву "Н". Колір синій.
7	Вивести на екран букву "Р". Колір синій.
8	Вивести на екран букву "V". Колір чорний.
9	Вивести на екран букву "W". Колір жовтий.
10	Вивести на екран букву "L". Колір синій.
11	Вивести на екран квадрат. Колір червоний.
12	Вивести на екран букву "N". Колір зелений.
13	Вивести на екран букву "U". Колір чорний.
14	Вивести на екран букву "V" .. Колір жовтий.
15	Вивести на екран букву "М" .. Колір чорний.
16	Вивести на екран букву "О". Колір синій.
17	Вивести на екран букву "К". Колір синій.
18	Вивести на екран букву "Q". Колір чорний.
19	Вивести на екран букву "X". Колір жовтий.
20	Вивести на екран букву "Z". Колір синій.

Контрольні запитання

1. Особливості малювання на Canvas у FireMonkey. Поясніть призначення BeginScene/EndScene.
2. Принципи обробки наджварних клавіш мобільного пристрою. Пояснить відмінність між OnKeyUp/OnKeyDown.
3. Особливості обробки жестів у Delphi.

3 ЛАБОРАТОРНА РОБОТА №3

Тема. Обчислення і побудова графіків простих математичних функцій. Робота з текстовими файлами. Використання зображень та повідомлень.

Мета. Провести обчислення та побудову графіків математичних функцій на мобільному пристрої. Навчитися працювати з текстовими файлами. Навчитися використовувати різні зображення в своїх проектах і виводити повідомлення, які сигналізують про закінчення будь-якого процесу.

Теоретичні відомості

Робота з масивами

Масив даних ініціалізується наступним чином:

type <ім'я масиву> = array [<число елементів>, <число елементів>,]
of <тип даних>;

Будь-масив може містити кілька вимірів [10], наприклад матриця a [i, j] містить два виміри, тобто адресація елементів ведеться по рядку j і одну j. Такий масив називається двовимірним. Приклад ініціалізації строкового одновимірного масиву даних, що складається з 10 елементів наведено нижче.

type myarr = array [1..10] of string;

Для використання нового типу даних (масиву) в програмі, потрібно оголосити змінну наступним чином.

var a: myarr;

Звернутися до конкретного елементу масиву можна наступним чином, наприклад:

a [1]: = 'My String';

Робота з файлами

Робота з файлами на мові Delphi здійснюється за допомогою використання файлової змінної (наприклад var f: textfile; змінна текстового файлу).

Для ініціалізації файлу служить наступна функція, наприклад: assignfile (f, '/mnt/sdcard/mysave.txt'); де '/mnt/sdcard/mysave.txt' шлях до файлу, який необхідно відкрити або створити (просто рядок), f -змінного текстового файлу (може мати будь-яке ім'я).

Далі необхідно відкрити файл або для запису, або для читання.

rewrite (f); або reset (f);

Важливо знати що команда rewrite повністю знищує всю інформацію в файлі, якщо він існує. Reset відкриває файл тільки для читання.

Для запису інформації в файл або читання її з файлу існують стандартні функції write (writeln - перехід на наступний рядок) і read (readln) відповідно. Наприклад writeln (f, a [1]); записує в файл елемент масиву, тобто рядок. Readln (f, a [1]); відповідно зчитує значення елемента з файлу.

Після запису або читання файл необхідно закрити. Для цього служить команда CloseFile (f).

Обробка виняткових ситуацій

Для обробки виняткових ситуацій в Delphi застосовується наступна конструкція.

```
try  
..... Оператори  
except on <Тип помилки> do  
..... оброблювач виключення  
end;
```

Типи виняткових ситуацій: помилка конвертації (EConvertError), розподіл на нуль (EZeroDivision), помилка введення-виведення (EInOutError) і т.д.

Наприклад

```
var i: integer;  
.....  
try  
  i := StrToInt (Edit1.Text);  
except on EConvertError do  
begin  
  ShowMessage ( 'Wrong Input !!!');  
  exit;  
end;  
end;
```

Даний фрагмент програмного коду демонструє перетворення рядка текстового поля в ціле число. Якщо функція StrToInt не може конвертувати рядок в число, викликається обробник виняткової ситуації EConvertError. З'являється повідомлення і команда exit здійснює вихід з обробника події.

Компонент TChart є дуже зручним для побудови різних діаграм, залежностей і графіків функції. Знаходиться він в палітрі компонентів TeeChart Lite > TChart (рис. 3.1).

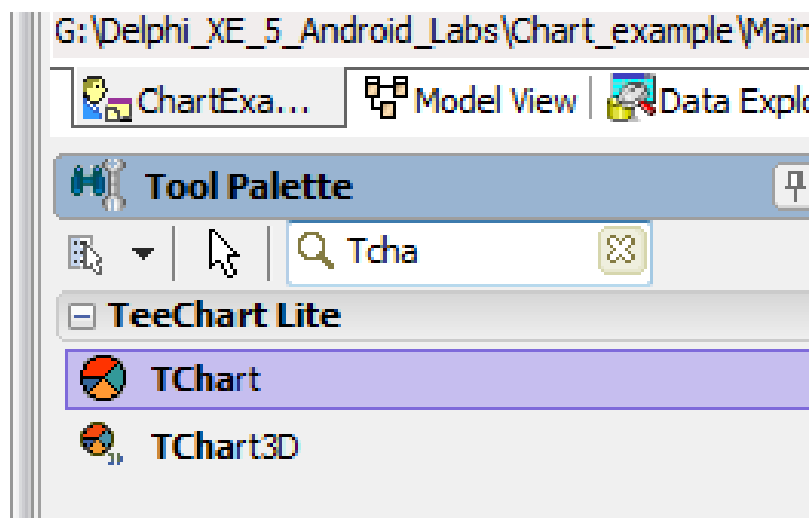


Рисунок 3.1 – Компонент TChart

При додаванні в проект компонент має вигляд (рис. 3.2).

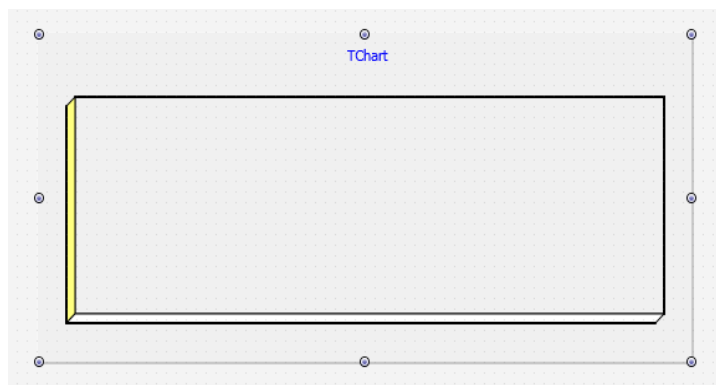


Рисунок 3.2 – Компонент TChart при додаванні на форму

Даний компонент спочатку не містить ніяких діаграм. Їх потрібно створити за допомогою подвійного клацання по компоненту. Створюємо графік у вигляді лінії (рис. 3.3).

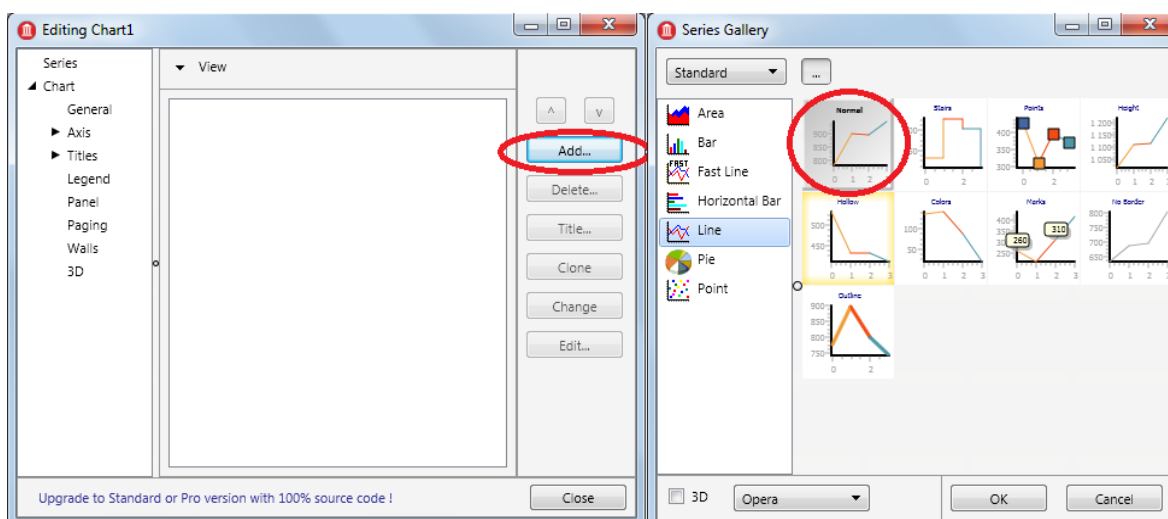


Рисунок 3.3 – Додавання графіків у компонент

Після натискання ОК маємо (рис. 3.4).

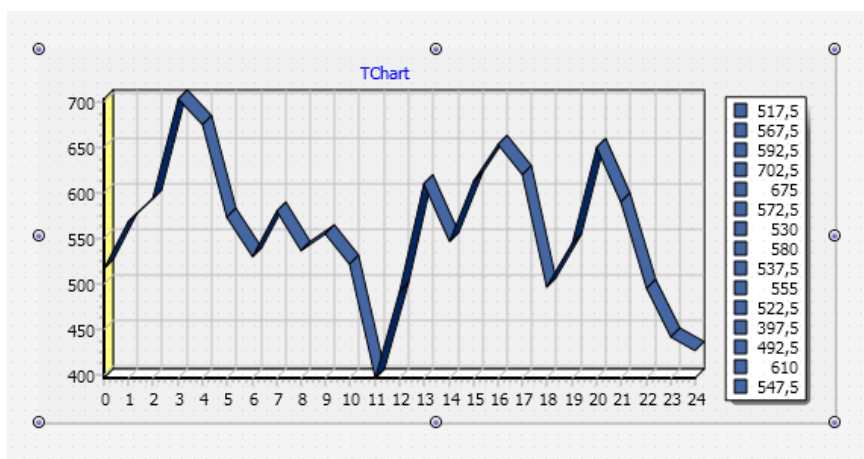


Рисунок 3.4 – Компонент TChart з одним графіком

Для того, щоб зробити графік одновимірним, потрібно зняти всі галочки в меню 3D (рис. 3.5).

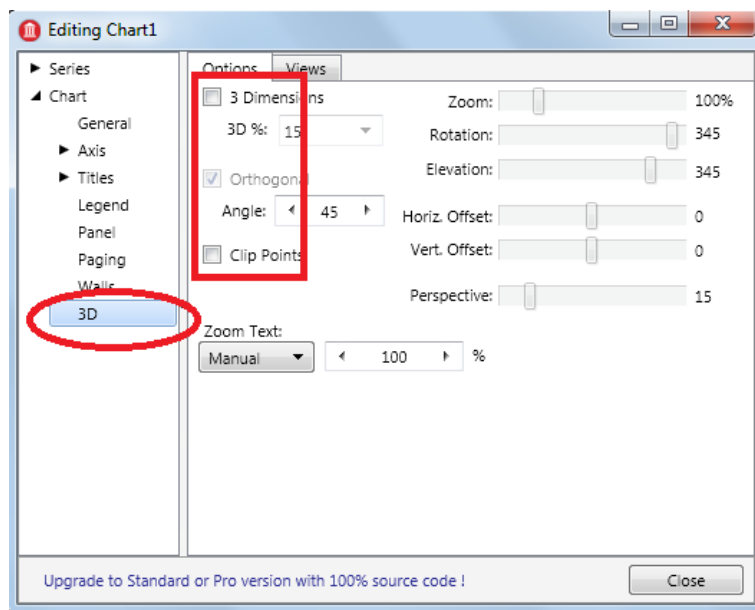


Рисунок 3.5 – Налаштування режиму 3D вигляду

При бажанні можна прибрати легенду (рис. 3.6).

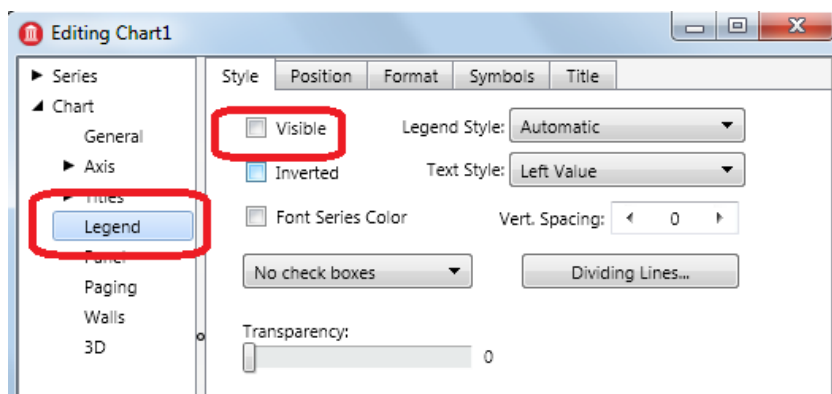


Рисунок 3.6 – Налаштування легенди

Можна також задати назву діаграми (рис. 3.7).

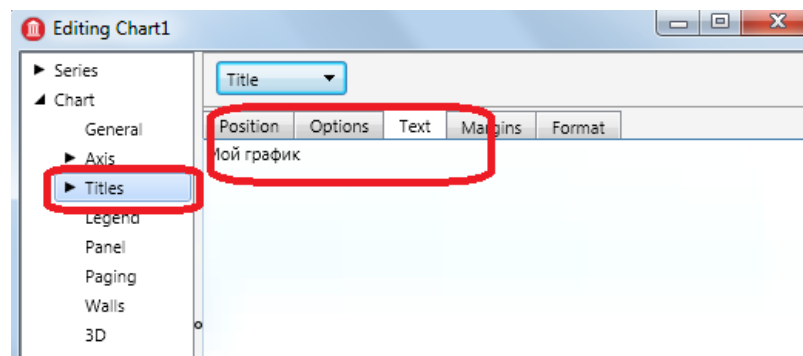


Рисунок 3.7 – Назва діаграми

Назва осей X і Y також задається наступним чином (рис. 3.8).

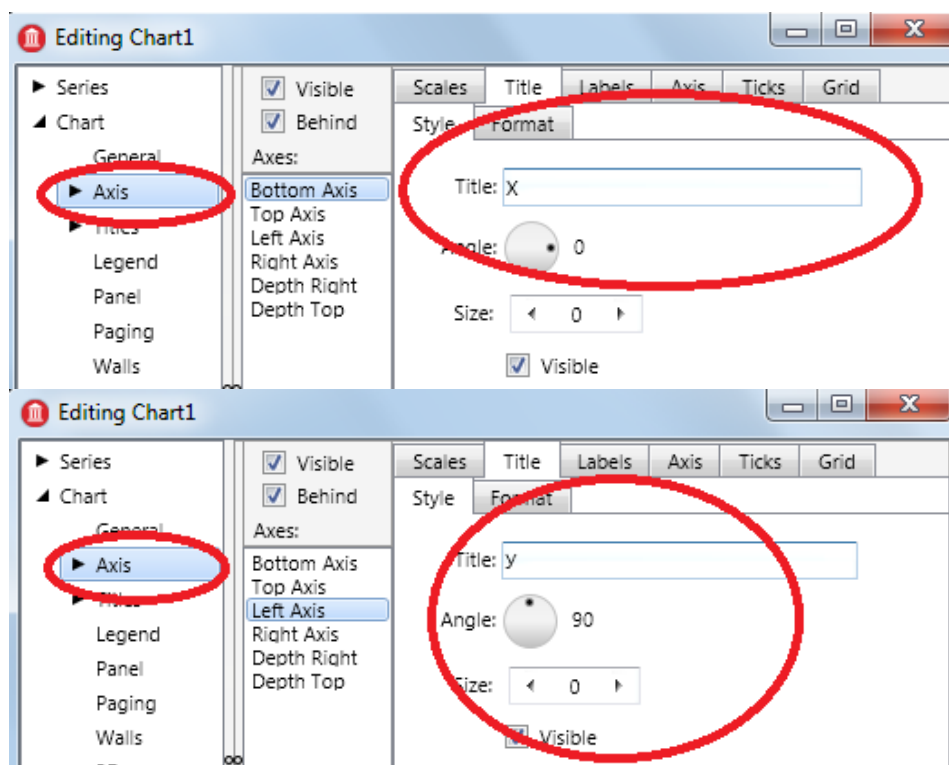


Рисунок 3.8 – Назва осей координат

Для налаштування товщини та кольору ліній графіка потрібно зробити наступні маніпуляції (рис. 3.9).

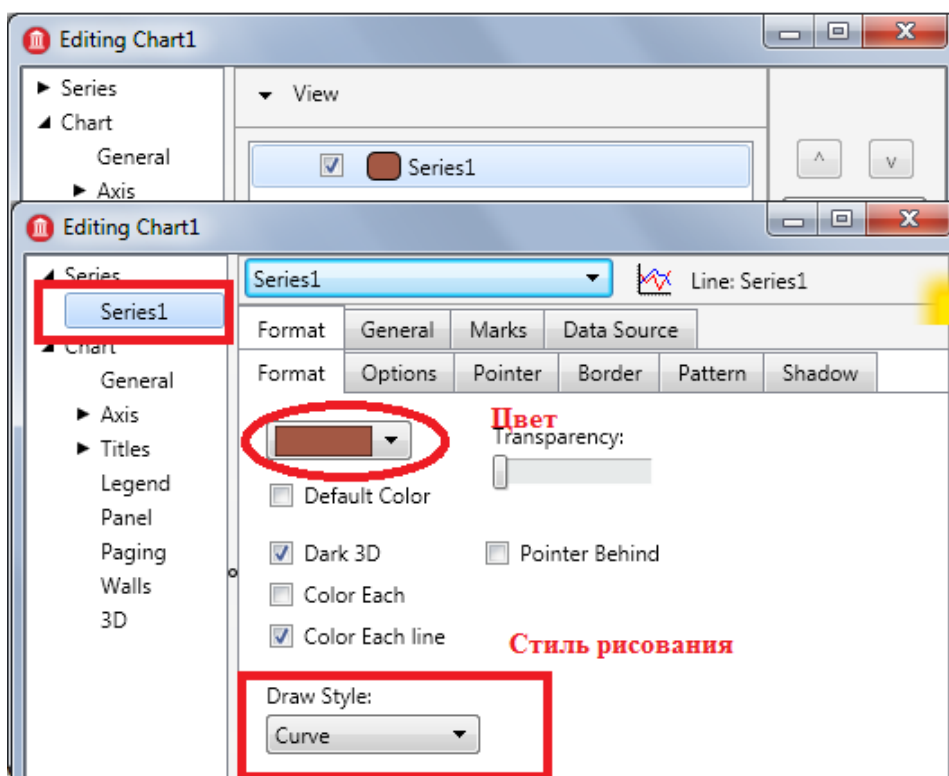


Рисунок 3.9 – Налаштування вигляду лінії графіку

Товщина лінії та її тип задається наступним чином (рис. 3.10).

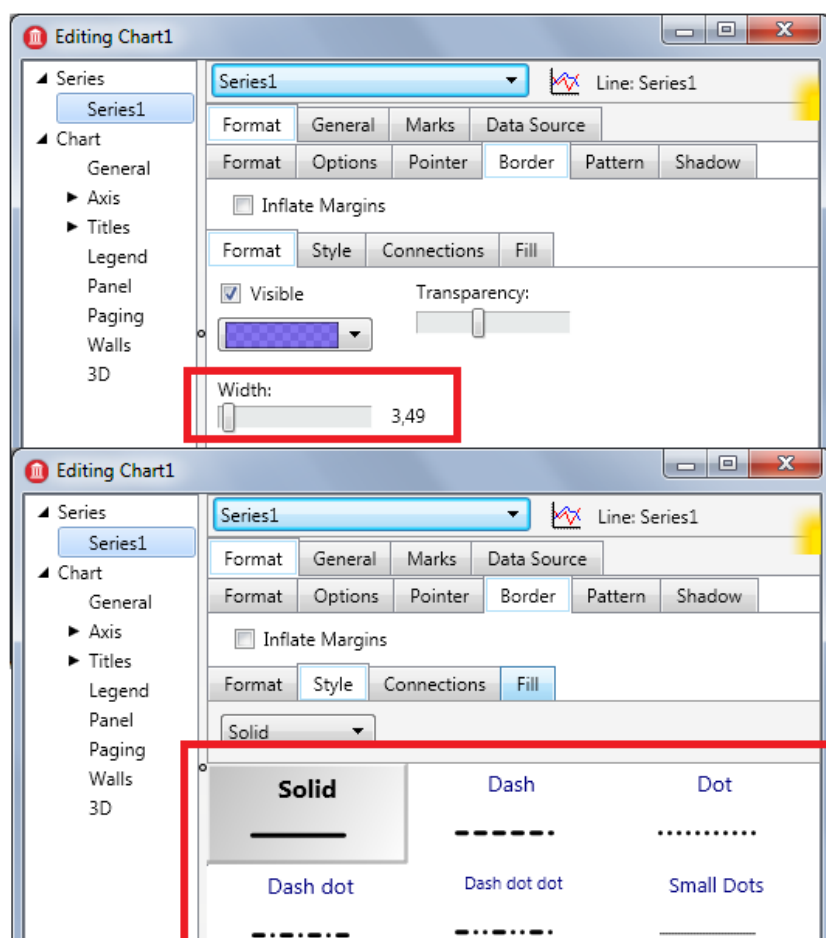


Рисунок 3.10 – Налаштування вигляду лінії графіку

Після зроблених налаштувань TChart має наступний вигляд (рис. 3.11).

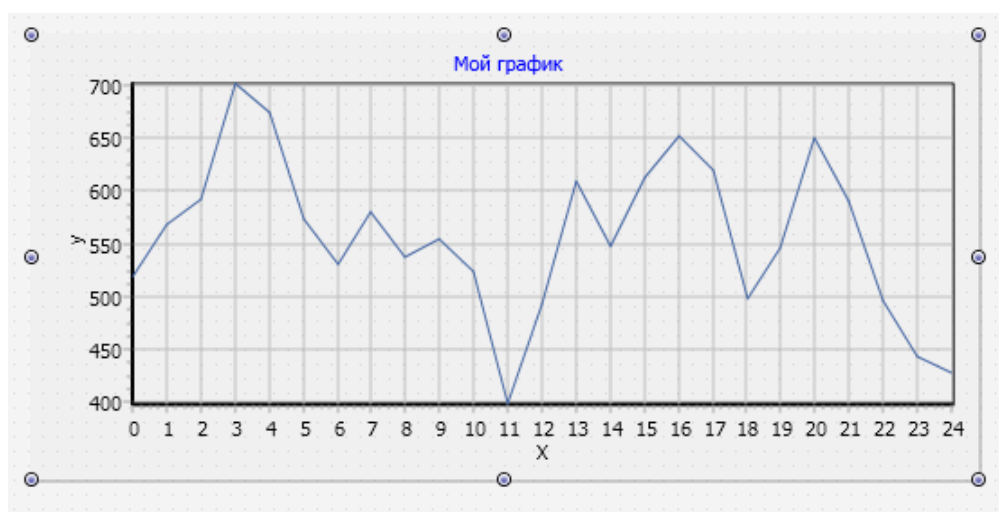


Рисунок 3.11 – Вигляд компонента TChart після налаштувань

Для відображення прогресу побудови графіка використовується компонент TProgressBar. Для виведення графіка функції використовується метод TChart AddXY (x, y). Нумерація графіків починається з нуля. (Series [0]).

Для відображення процесу побудови графіка використовується Application.ProcessMessages.

Пошук файлів в директорії

Пошук файлів здійснюється з використанням змінної типу TSearchRec. Список знайдених файлів виводиться в компонент TListBox. Стандартна функція FindFirst (<шлях пошуку ", " тип файлів, які потрібно знайти ", " файлова змінна пошуку файлів>) здійснює пошук першого файлу, який відповідає бажаним налаштувань. Наприклад потрібно знайти файли в директорії / mnt / sdcard / с розширенням * .pdf. Тоді функція буде мати вигляд

*FindFirst (' / mnt / sdcard / *. Pdf', faAnyFile, f);*

faAnyFile - в даному випадку проводиться пошук будь-якого файлу (може приймати значення faSysFile - системні файли, faDirectory - тільки папки і т.д.). Опис всіх змінних в модулі System.SysUtils. f - поточна файлова змінна.

Функція FindNext (<змінна пошуку файлів>) знаходить всі інші файли, що задовольняють умові FindFirst.

Повідомлення на мобільних пристроях

Для роботи з повідомленнями потрібно використовувати компонент NotificationCenter. Даний компонент є не візуальним, тобто під час роботи програми він відображатися не буде. Приклад програмного коду виведення повідомлення після закінчення процесу побудови графіка функції.

```
var Notification: TNotification;  
.....  
// Виведення повідомлення  
if NotificationCenter1.Supported then // якщо повідомлення підтримуються пристро-  
єм - виводимо  
begin  
    Notification := NotificationCenter1.CreateNotification; // Створюємо повідомлення  
    try  
        Notification.Name := 'Lab4 '; // Тема повідомлення  
        Notification.AlertBody := 'Графік побудований!'; // Тіло повідомлення  
        Notification.FireDate := Now + EncodeTime (0,0,5,0); // Час, через яке буде показ  
повідомлення  
        NotificationCenter1.ScheduleNotification (Notification); // виконання показу  
повідомлення  
    finally  
        Notification.DisposeOf; // прибираємо за собою  
    end;  
end;
```

Якщо необхідно видалити виведене повідомлення, можна це зробити в такий спосіб

```
if NotificationCenter1.Supported then  
begin
```



```
NotificationCenter1.CancelNotification ( 'Lab4');
end;
```

Компонент `TrackBar` дуже зручний для установки будь-яких налаштувань або динамічного зміни значення будь-якого параметра (рис. 3.12).

Рисунок 3.12 – Компонент `TrackBar`

Містить наступні корисні властивості: `Max`, `Min`, `Value` (Максимальне значення, мінімальне значення, поточне значення).

Компонент `ComboEdit` (поле зі списком) дуже зручний для вибору будь-якого файлу, так як може містити в собі весь список доступних файлів на пристрої.

Хід роботи

Зовнішній вигляд інтерфейсу для розрахунку і виведення графіків функцій зображений на рис. 3.13.

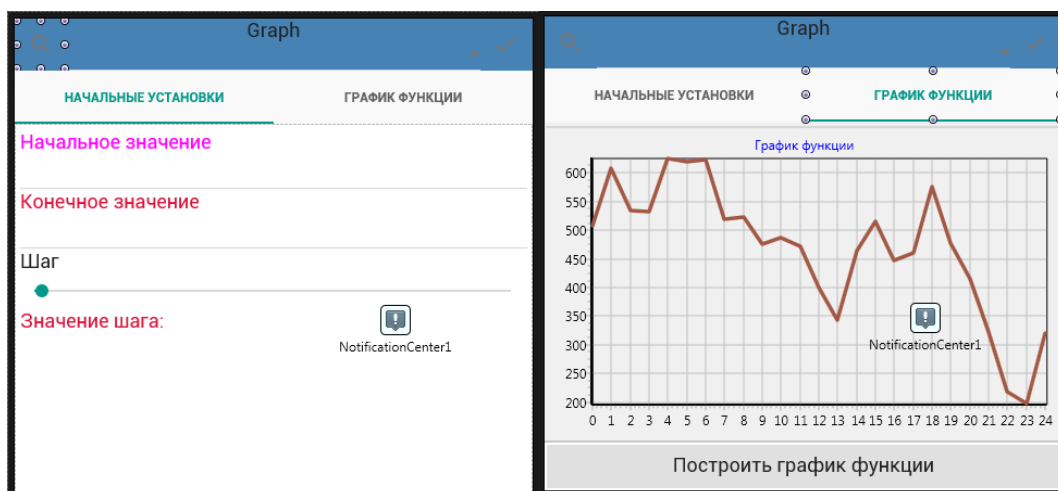


Рисунок 3.13 – Зовнішній вигляд інтерфейсу розробленого додатку

Лістинг програми

```
unit Main;
interface
uses
  System.SysUtils, System.Types, System.UITypes, System.Classes, System.Variants,
  FMX.Types, FMX.Controls, FMX.Forms, FMX.Graphics, FMX.Dialogs, FMX.StdCtrls,
  FMXTee.Engine, FMXTee.Procs, FMXTee.Chart, FMX.TabControl, FMX.Edit,
  FMX.Objects, FMX.Controls.Presentation, FMXTee.Series, FMX.ListBox,
  FMX.Notification, Math, FMX.ComboEdit;
type
  TMainForm = class (TForm)
    ToolBar1: TToolBar;
    Label1: TLabel;
    Image1: TImage;
```

```

Label2: TLabel;
Edit1: TEdit;
Label3: TLabel;
Edit2: TEdit;
Label4: TLabel;
TabControl1: TTabControl;
TabItem1: TTabItem;
TabItem2: TTabItem;
Chart1: TChart;
TrackBar1: TTrackBar;
Label5: TLabel;
Series1: TLineSeries;
Button3: TButton;
Button1: TButton;
Edit3: TEdit;
Button4: TButton;
ProgressBar1: TProgressBar;
NotificationCenter1: TNotificationCenter;
ComboEdit1: TComboEdit;
procedure TrackBar1Change (Sender: TObject);
procedure Button1Click (Sender: TObject);
procedure Button4Click (Sender: TObject);
procedure FormShow (Sender: TObject);
procedure Button3Click (Sender: TObject);
private
  {Private declarations}
public
  {Public declarations}
end;
type MyOpt = array [1..3] of Double;
var
  MainForm: TMainForm;
  step: real;
  xn, xk, y: real;
  Options: MyOpt;
implementation
{$ R * .fmx}
// Наша функція
function myf (x: real): real;
begin
  if (x <= 0) then myf: = 2
  else if (x > 0) and (x < 20) then myf: = 10 * exp (-0.08 * x) * sin (x)
  else myf: = 5;
end;
procedure TMainForm.Button1Click (Sender: TObject);
var f: textfile; i: integer;
begin
  // Завантаження зображення з карти пам'яті
  Image1.Bitmap.LoadFromFile ( '/' / mnt / sdcard /' + ComboEdit1.Text);

```

```

// Збереження налаштувань початкових значень в текстовий файл
Options [1]: = xn;
Options [2]: = xk;
Options [3]: = step;
assignfile (f, '/ mnt / sdcard / myopt_lab4.txt');
rewrite (f);
  for i: = 1 to 3 do
  begin
    writeln (f, Options [i]);
  end;
closefile (f);
ShowMessage ( 'Налаштування збережені в файл!');
end;
// Пошук файлів зображень на карті пам'яті
procedure TMainForm.Button3Click (Sender: TObject);
var f: tsearchrec;
begin
  if findfirst ( '/ mnt / sdcard / *. jpg', faanyfile, f) <> 0 then exit;
  ComboEdit1.Items.Add (f.name);
  while findnext (f) = 0 do
  begin
    ComboEdit1.Items.Add (f.Name);
  end;
  findclose (f);
end;
procedure TMainForm.Button4Click (Sender: TObject);
var Notification: TNotification; x: real;
begin
  Chart1.Series [0] .Clear;
// Введення даних //
try
  xn: = StrToFloat (Edit1.Text);
except on EConvertError do
  begin
    ShowMessage ( 'Неправильне початкове значення!');
    Exit;
  end;
end;
try
  xk: = StrToFloat (Edit2.Text);
except on EConvertError do
  begin
    ShowMessage ( 'Неправильне кінцеве значення!');
    Exit;
  end;
end;
x: = xn;
ProgressBar1.Min: = xn;
Progressbar1.Max: = xk;

```

```

while (xk> x) do
begin
  Application.ProcessMessages;
  y:= myf (x);
  x:= x + step;
  Chart1.Series [0] .AddXY (x, y);
  ProgressBar1.Value:= x;
end;
// Висновок повідомлення
if NotificationCenter1.Supported then
begin
  Notification:= NotificationCenter1.CreateNotification;
  try
    Notification.Name:='Lab4 ';
    Notification.AlertBody:= 'Графік побудований!';
    Notification.FireDate:= Now + EncodeTime (0,0,5,0);
    NotificationCenter1.ScheduleNotification (Notification);
  finally
    Notification.DisposeOf;
  end;
end;
end;
procedure TMainForm.FormShow (Sender: TObject);
var f: textfile; i: integer;
begin
  if FileExists ( '/ mnt / sdcard / myopt_lab4.txt') then
  begin
    assignfile (f, '/ mnt / sdcard / myopt_lab4.txt');
    reset (f);
    for i:= 1 to 3 do
    begin
      readln (f, Options [i]);
    end;
    Edit1.Text:= Options [1] .ToString ();
    Edit2.Text:= Options [2] .ToString ();
    TrackBar1.Value:= Options [3];
    closefile (f);
  end;
end;
procedure TMainForm.TrackBar1Change (Sender: TObject);
begin
  Label5.Text:= 'Значення кроку' + FloatToStr (Trackbar1.Value);
  step:= Trackbar1.Value;
end; end.

```

Індивідуальне завдання

На основі програмного коду наведеної лабораторної роботи виконати завдання відповідно до варіанту за списком (табл. 3.1).

Таблиця 3.1 – Варіанти індивідуальних завдань

№ Варіанта	Індивідуальне завдання
1	Вивести графік функції $y = 2 \cdot \sin(x)$ на інтервалі $[-50..50]$ з кроком 0.1. На інтервалі $[50..70]$ $y = 4 \cdot \cos(x)$
2	Вивести графік функції $y = x^3$ на інтервалі $[0..10]$ з кроком 0.1. На інтервалі $[-10..0]$ $y = e^2 \cdot x$
3	Вивести графік функції $y = 6 \cdot \sin(x) \cdot x$ на інтервалі $[0..50]$ з кроком 0.1. На інтервалі $[50..80]$ $y = x$
4	Вивести графік функції $y = 2 \cdot \sin(x) \cdot x^2$ на інтервалі $[-10..10]$ з кроком 0.1. На інтервалі $[10..20]$ $y = x^3$
5	Вивести графік функції $y = 2 \cdot x^2$ на інтервалі $[-5..10]$ з кроком 0.1. На інтервалі $[10..30]$ $y = x^3$
6	Вивести графік функції $y = 2$ на інтервалі $[-5..0]$ з кроком 0.1. На інтервалі $[0..20]$ $y = \sin(x)$
7	Вивести графік функції $y = 2 \cdot \sin(x)$ на інтервалі $[-2..6]$ з кроком 0.1. На інтервалі $[10..20]$ $y = x^2$
8	Вивести графік функції $y = \sin(x) \cdot x$ на інтервалі $[-0..10]$ з кроком 0.1. На інтервалі $[10..15]$ $y = x$
9	Вивести графік функції $y = 8 \cdot x^2$ на інтервалі $[-10..10]$ з кроком 0.1. На інтервалі $[10..20]$ $y = 9$
10	Вивести графік функції $y = 8 \cdot \sin(x)$ на інтервалі $[-2..2]$ з кроком 0.1. На інтервалі $[2..8]$ $y = 2 \cdot x^2$
11	Вивести графік функції $y = \sin(x)$ на інтервалі $[-1..4]$ з кроком 0.1. На інтервалі $[4..8]$ $y = 5 \cdot x$
12	Вивести графік функції $y = 8 \cdot \sin(x)$ на інтервалі $[-10..0]$ з кроком 0.1. На інтервалі $[0..8]$ $y = 2 \cdot x^2$
13	Вивести графік функції $y = 8 \cdot \sin(x)$ на інтервалі $[-1..3]$ з кроком 0.1. На інтервалі $[3..6]$ $y = 2 \cdot x^2$
14	Вивести графік функції $y = 2 \cdot \sin(x) \cdot \cos(x)$ на інтервалі $[-4..0]$ з кроком 0.1. На інтервалі $[0..4]$ $y = x^2$
15	Вивести графік функції $y = \sin(x)$ на інтервалі $[-0..3]$ з кроком 0.1. На інтервалі $[3..8]$ $y = x^2$

Контрольні питання

1. Який клас використовується для виведення повідомлень на мобільному пристрою?
2. Який компонент слугує для виведення графіків? Наведіть приклад.
3. Як встановити зображення у якості фону? Який клас використовується для виведення зображень?

4 ЛАБОРАТОРНА РОБОТА №4

Тема. Робота з таймером, використання вібрації і вбудованого спалаху мобільного пристрою. Використання акселерометра. Використання стилів в мобільних додатках.

Мета. Навчитися працювати з класом таймер при програмуванні на мобільні пристрої, використовувати вібрацію в додатках. Освоїти роботу з вбудованим спалахом мобільного пристрою і розібратися з роботою акселерометра. Навчитися застосовувати різні стилі оформлення мобільного додатка.

Теоретичні відомості

Клас TTimer

Компонент Delphi TTimer призначений для відліку часу. Він знаходиться в палітрі System> TTimer. Має єдина подія OnTimer. Після закінчення певного інтервалу часу викликається обробник. Має властивості Enabled (включений / виключений), Interval (час в мілісекундах) за замовчуванням 1000 (1 сек.).

Даний компонент дуже зручний для виведення системного часу. Наприклад фрагмент програмного коду показує поточний час кожну секунду

```
var t: TDateTime; // повідомляємо змінну для виведення поточного часу.
```

При створенні форми (FormCreate) додаємо наступний код.

```
.....  
Timer1.Enabled := true; // включаємо таймер  
Timer1.Interval := 1000; // інтервал 1 сек.
```

При спрацьовуванні таймера (подія OnTimer) виводиться поточний час.

```
.....  
t := time;  
Label1.Text := TimeToStr(t);
```

Таким чином, кожну секунду буде виводитися даний час в Label1 наприклад.

Клас TMotionSensor

Датчик руху дозволяє отримати інформацію про зміну орієнтації та рух пристрою в просторі в трьох площинах, уздовж прив'язаних до пристрою осей X, Y, Z.

Ось X спрямована вздовж екрану пристрою, вісь Y - від низу до верху. Вісь Z перпендикулярно осі пристрою.

Доступні властивості AccelerationX, AccelerationY, AccelerationZ містять значення вимірюваного в Галах поточного прискорення (1 Гал 9.8 м / с²).

Клас TStyleBook

Даний компонент служить для завдання стилю мобільного додатка. Сучасні мобільні додатки мають різні стилі оформлення, для того, щоб залучити користувачів. FireMonkey містить в собі клас TStyleBook (рис. 4.1).

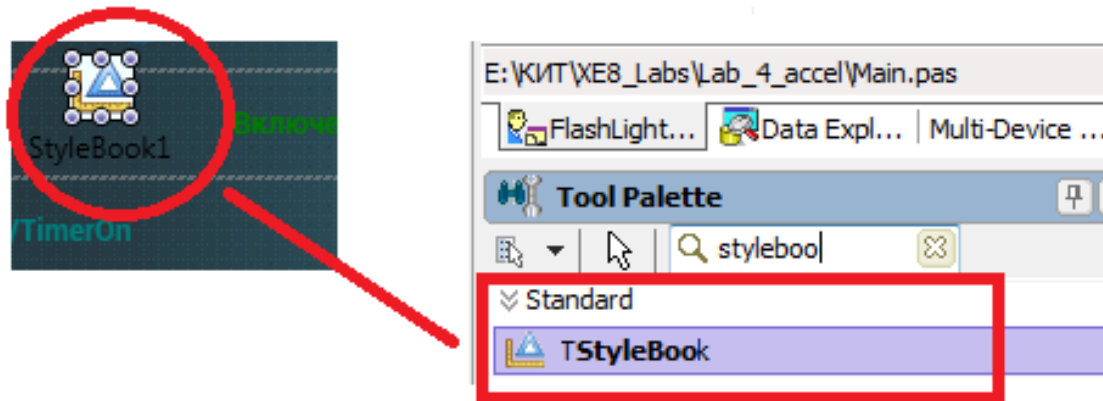


Рисунок 4.1 – Компонент StyleBook

Подвійним кліком по компоненту StyleBook можна завантажити заздалегідь заготовлені стилі оформлення мобільного додатка (рис. 4.2).

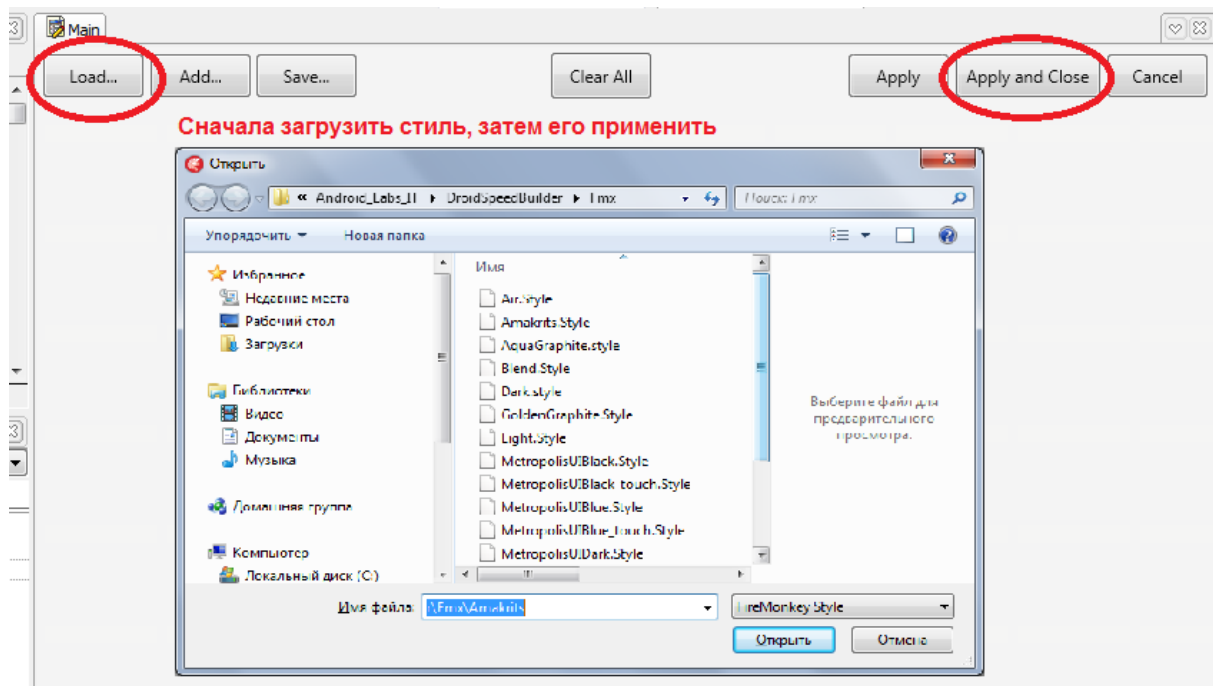


Рисунок 4.2 – Зовнішній вигляд інтерфейсу розробленого додатку

Для встановлення обраного стилю для активності (форми) додатка потрібно виконати наступні дії (рис. 4.3).

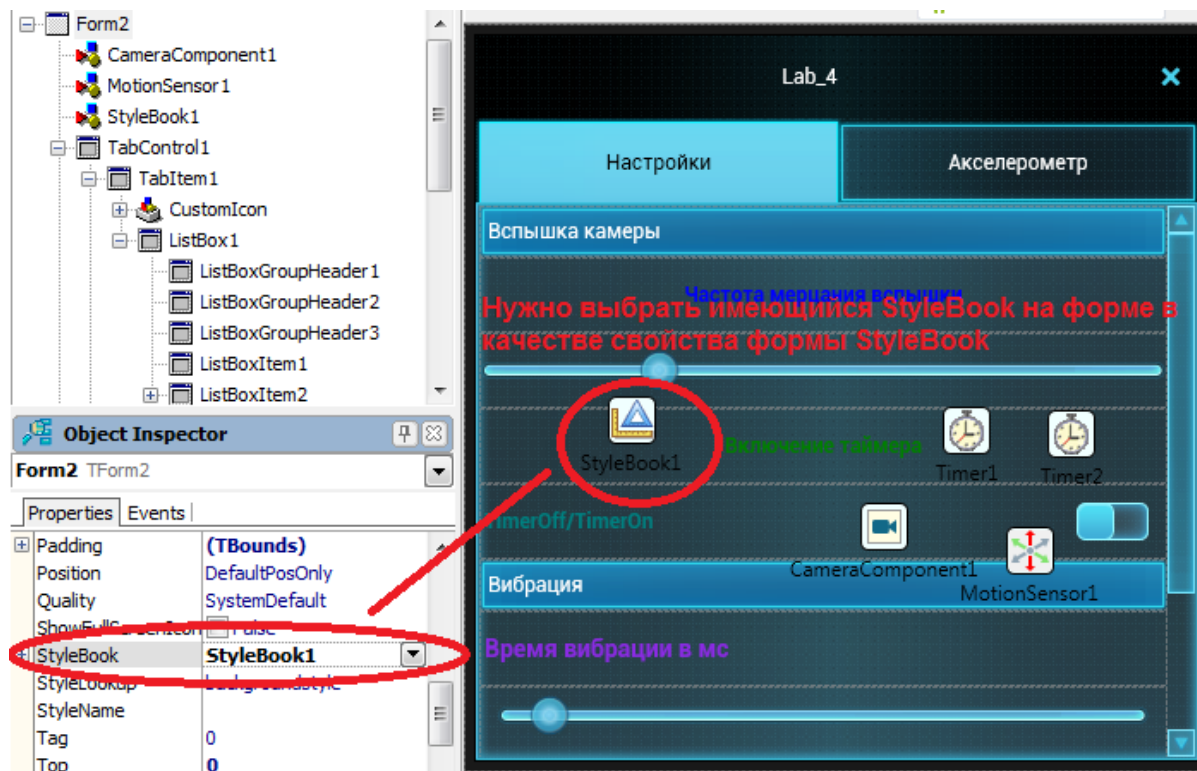


Рисунок 4.3 – Встановлення обраного стилю

Для задання стилю інших форм (якщо є) потрібно виконати аналогічні дії.

Клас *TCameraComponent*

Даний клас виконаний у вигляді стандартного компонента. Призначений для роботи з вбудованою камерою мобільного пристрою. У даній лабораторній роботі він буде використовуватися для доступу і управління спалахом камери, якщо вона є.

Містить властивість *Active* (*true*, *false*). За допомогою цієї властивості можна включати або вимикати даний компонент.

Наведено фрагмент програмного коду, який дозволяє включити або вимкнути спалах камери.

```
procedure TMyForm.SetFlashlightState (Active: Boolean);
begin
  if Active then
  begin
    CameraComponent1.TorchMode: = TTorchMode.tmModeOn; // увімкнути
  end else
  begin
    CameraComponent1.TorchMode: = TTorchMode.tmModeOff; // вимкнути
  end;
end;
```

CameraComponent має одну подію *OnSampleBufferReady*.

Використовуючи обробник цієї події можна, наприклад з використанням потоку відображати захоплене зображення з камери в компоненті *Image*.


```

procedure TForm2.CameraComponent1SampleBufferReady (Sender: TObject;
  const ATime: TMediaTime);
begin
  TThread.Synchronize (TThread.CurrentThread,
    procedure
    begin
      // Відображення захоплення камери в Image
      CameraComponent1.SampleBufferToBitmap (Image1.Bitmap, True);
    end);
end;

```

Таким чином представлений вище код здійснить динамічне виведення зображення, захоплене камерою в Image. Тобто ми отримаємо ефект дзеркала. За допомогою CameraComponent можна вибрати або використання фронтальної камери, або задньої.

```

// Вибір задньої камери
CameraComponent1.Kind: = TCameraKind.BackCamera;
// Вибір фронтальної камери
CameraComponent1.Kind: = TCameraKind.FrontCamera;

```

Використання вібрації

Для включення можливості вібрації на мобільному пристрої потрібно використовувати Java класи [11,12,13]. Для їх використання в Delphi вже є обгортки AndroidApi (файли з розширенням .pas). Їх потрібно підключити в uses (рис.4.4).

```

uses
  System.SysUtils, System.Types, System.UITypes, System.Classes, System.Variants,
  FMX.Types, FMX.Controls, FMX.Forms, FMX.Graphics, FMX.Dialogs, FMX.StdCtrls,
  FMX.Media, FMX.Controls.Presentation,
  // Androidapi.JNI.Bridge,
  Androidapi.Helpers, Androidapi.JNI.Os,
  // Androidapi.FMX.JNI,
  // Androidapi.JNI.Media,
  FMX.Platform.Android,
  Androidapi.JNI.GraphicsContentViewText,
  // Androidapi.FMX.JNI.Types,

```

Рисунок 4.4 – Підключення нативних Java класів

Приклад здійснення вібрації певної тривалості

```

procedure TMyForm.Button1Click (Sender: TObject);
var i: integer; Vibrator: JVibrator;
begin
  // Ініціалізація класу здійснення вібрації
  Vibrator: = TJVibrator.Wrap (SharedActivity.
    getSystemService (TJContext.JavaClass.VIBRATOR_SERVICE));

```

```

    if Vibrator.hasVibrator () then // якщо вібрація підтримується
    // Власне Вібруючи
        Vibrator.vibrate (Round (TrackBar2.Value)); // За допомогою Trackbar задаємо тривалість вібрації в мс.
    end;

```

Інтерфейс розробленого додатка наведено на рис. 4.5.

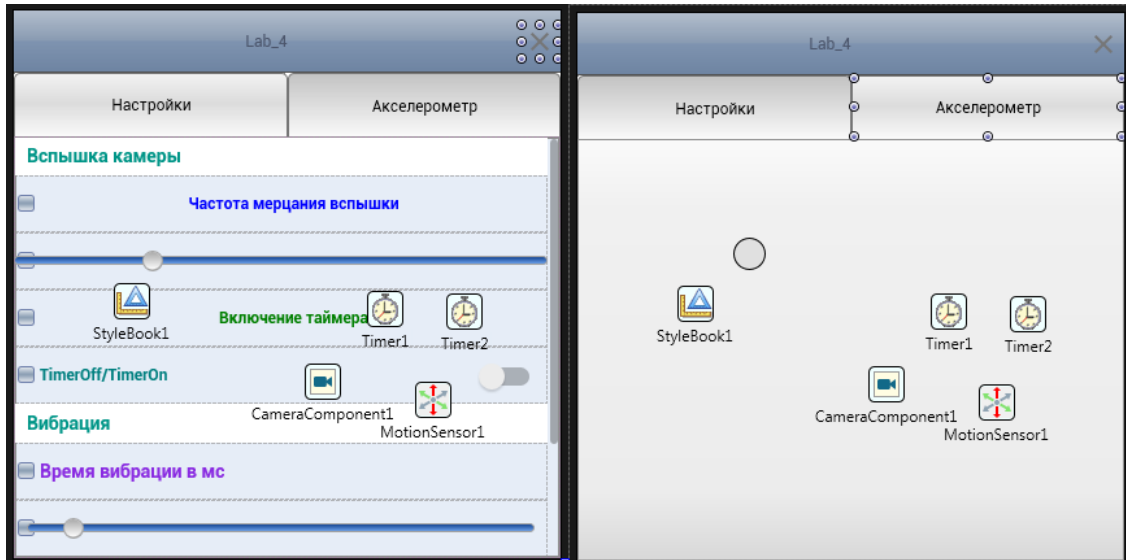


Рисунок 4.5 – Інтерфейс демонстраційного додатку

Для завершення роботи програми використовується Java клас MainActivity з FMX.Platform.Android.

```

procedure TMyForm.Button2Click (Sender: TObject);
begin
    MainActivity.Finish; // Завершення роботи головною активності (тобто всього
                        // додатку)
end;

```

Для демонстрації роботи акселерометра, використовується компонент TCircle (Shapes). Акселерометр буде управляти положенням Circle щодо PaintBox (PaintBox - контейнер (Parent) для Circle).

Для відображення прискорень використовується компонент TLabel.

Опитування стану прискорень акселерометра TMotionSensor буде проводитися за таймером [14].

```

procedure TMyForm.Timer2Timer (Sender: TObject);
begin
    // Включити датчик руху
    MotionSensor1.Sensor.Start;
    // Позиція кола згідно з показаннями датчика
    Circle1.Position.X := MotionSensor1.Sensor.AccelerationX * 100 + PaintBox1.Width / 2;
    Circle1.Position.Y := MotionSensor1.Sensor.AccelerationY * 100 + PaintBox1.Height / 2;

```

```
// Обмеження, щоб коло далеко не пішло за рамки форми
if Circle1.Position.X >= PaintBox1.Width then Circle1.Position.X := PaintBox1.Width / 2;
if Circle1.Position.Y >= PaintBox1.Height then Circle1.Position.Y := PaintBox1.Height / 2;
if Circle1.Position.X <= 0 then Circle1.Position.X := PaintBox1.Width / 2;
if Circle1.Position.Y <= 0 then Circle1.Position.Y := PaintBox1.Height / 2;
// Відобразити координати датчика
Label2.Text := 'X=' + FloatToStrf (MotionSensor1.Sensor.AccelerationX, ffGeneral, 2,5);
Label3.Text := 'Y=' + FloatToStrf (MotionSensor1.Sensor.AccelerationY, ffGeneral, 2,5);
end;
```

Параметр ffGeneral дозволяє налаштовувати кількість знаків після коми (в дійсних числах).

При спрацьовуванні таймера Timer1 (якщо активний) будемо проводити перемикання стану спалаху камери (вкл / викл). Частота перемикань спалаху буде залежати від періоду таймера (Timer1.Interval).

```
// миготіння спалахом
procedure TMyForm.Timer1Timer (Sender: TObject);
begin
if flag then
begin
SetFlashlightState (True); // функція перемикання стану спалаху
flag := false;
end
else
begin
SetFlashlightState (False);
flag := true;
end;
end;
```

Змінювати період таймера будемо динамічно за допомогою Trackbar (Подія OnChange).

```
// Налаштування періоду таймера
procedure TMyForm.TrackBar1Change (Sender: TObject);
begin
Timer1.Interval := Round (Trackbar1.Value); // Задаємо період таймера
end;
```

Лістинг програми

```
unit Main;
interface
uses
System.SysUtils, System.Types, System.UITypes, System.Classes, System.Variants,
FMX.Types, FMX.Controls, FMX.Forms, FMX.Graphics, FMX.Dialogs,
FMX.StdCtrls,
```

```

    FMX.Media, FMX.Controls.Presentation,
    // Androidapi.JNIBridge,
    Androidapi.Helpers, Androidapi.JNI.Os,
    // Androidapi.JNI.App,
    // Androidapi.JNI.Media,
    FMX.Platform.Android,
    Androidapi.JNI.GraphicsContentViewText,
    // Androidapi.JNI.JavaTypes,
    FMX.ListBox, FMX.Layouts, System.Sensors,
    FMX.Objects, FMX.TabControl, System.Sensors.Components;
type
    TMyForm = class (TForm)
        ToolBar1: TToolBar;
        CameraComponent1: TCameraComponent;
        Switch1: TSwitch;
        Label1: TLabel;
        TrackBar1: TTrackBar;
        Timer1: TTimer;
        Button1: TButton;
        ListBox1: TListBox;
        ListBoxGroupHeader1: TListBoxGroupHeader;
        ListBoxItem1: TListBoxItem;
        ListBoxItem2: TListBoxItem;
        ListBoxItem3: TListBoxItem;
        ListBoxItem4: TListBoxItem;
        ListBoxGroupHeader2: TListBoxGroupHeader;
        ListBoxItem5: TListBoxItem;
        ListBoxItem6: TListBoxItem;
        ListBoxItem7: TListBoxItem;
        TrackBar2: TTrackBar;
        ListBoxGroupHeader3: TListBoxGroupHeader;
        ListBoxItem8: TListBoxItem;
        MotionSensor1: TMotionSensor;
        TabControl1: TTabControl;
        TabItem1: TTabItem;
        TabItem2: TTabItem;
        PaintBox1: TPaintBox;
        Circle1: TCircle;
        Timer2: TTimer;
        Label2: TLabel;
        Button2: TButton;
        Label3: TLabel;
        StyleBook1: TStyleBook;
        Switch2: TSwitch;
        procedure Switch1Switch (Sender: TObject);
        procedure Timer1Timer (Sender: TObject);
        procedure FormShow (Sender: TObject);
        procedure Button1Click (Sender: TObject);
        procedure Switch2Switch (Sender: TObject);
    end;

```

```

    procedure Timer2Timer (Sender: TObject);
    procedure TrackBar1Change (Sender: TObject);
    procedure Button2Click (Sender: TObject);
private
    {Private declarations}
    procedure SetFlashlightState (Active: Boolean);
public
    {Public declarations}
end;
var
    MyForm: TMyForm;
    flag: boolean;
implementation
{$ R * .fmx}
{$ R * .iPhone4in.fmx IOS}
{$ R * .LgXhdpiPh.fmx ANDROID}
{$ R * .GGlass.fmx ANDROID}
procedure TMyForm.Button1Click (Sender: TObject);
var i: integer; Vibrator: JVibrator;
begin
    Vibrator: = TJVibrator.Wrap (SharedActivity.getSystemService (TJCon-
text.JavaClass.VIBRATOR_SERVICE));
    if Vibrator.hasVibrator () then
        Vibrator.vibrate (Round (TrackBar2.Value));
end;
// вихід з програми
procedure TMyForm.Button2Click (Sender: TObject);
begin
    MainActivity.Finish;
end;
procedure TMyForm.FormShow (Sender: TObject);
begin
    flag: = true;
    CameraComponent1.Active:=false;
end;
// включити-вимкнути підсвічування
procedure TMyForm.SetFlashlightState (Active: Boolean);
begin
    if Active then
        begin
            CameraComponent1.TorchMode: = TTorchMode.tmModeOn;
        end else
            CameraComponent1.TorchMode: = TTorchMode.tmModeOff;
end;
// включати вимикати спалах (таймер і компонент камери)
procedure TMyForm.Switch1Switch (Sender: TObject);
begin
    if Switch1.IsChecked then

```

```

begin
    CameraComponent1.Active:=true;
    Timer1.Enabled: = true;
end

else
begin
    Timer1.Enabled: = false;
    CameraComponent1.Active:=false;
end;
end;
// включити вимкнути акселерометр
procedure TMyForm.Switch2Switch (Sender: TObject);
begin
    if Switch2.IsChecked then
    begin
        MotionSensor1.Active:=not MotionSensor1.Active;
        Timer2.Enabled: = not Timer2.Enabled;
    end;
end;
// миготіння підсвічуванням
procedure TMyForm.Timer1Timer (Sender: TObject);
begin
    if flag then
    begin
        SetFlashlightState (True);
        flag: = false;
    end
    else
    begin
        SetFlashlightState (False);
        flag: = true;
    end;
end;
procedure TMyForm.Timer2Timer (Sender: TObject);
begin
    // Включити датчик руху
    MotionSensor1.Sensor.Start;
    // Позиція кола згідно з показаннями датчика
    Circle1.Position.X: = MotionSensor1.Sensor.AccelerationX * 100 + Paint-
Box1.Width / 2;
    Circle1.Position.Y: = MotionSensor1.Sensor.AccelerationY * 100 + Paint-
Box1.Height / 2;
    // Обмеження, щоб коло далеко не пішов за рамки форми
    if Circle1.Position.X> = PaintBox1.Width then Circle1.Position.X: = Paint-
Box1.Width / 2;
    if Circle1.Position.Y> = PaintBox1.Height then Circle1.Position.Y: = Paint-
Box1.Height / 2;

```

```

if Circle1.Position.X <= 0 then Circle1.Position.X: = PaintBox1.Width / 2;
if Circle1.Position.Y <= 0 then Circle1.Position.Y: = PaintBox1.Height / 2;
// Відобразити координати датчика
Label2.Text: = 'X =' + FloatToStrf (MotionSensor1.Sensor.AccelerationX, ffGeneral,
2,5);
Label3.Text: = 'Y =' + FloatToStrf (MotionSensor1.Sensor.AccelerationY, ffGeneral,
2,5);
end;
// встановлює час таймера
procedure TMyForm.TrackBar1Change (Sender: TObject);
begin
    Timer1.Interval: = Round (Trackbar1.Value);
end;
end.

```

Індивідуальне завдання

Скомпілювати наведений як приклад програмний код і встановити арк файл на мобільному пристрої. Перевірити функціональність мобільного додатка. У наведеному прикладі зробити наступне: при спрацьовуванні таймера здійснити переміщення Circle по горизонтальній осі.

При виході за межі PaintBox здійснити в коді повернення назад Circle. При зрушуванні Switch (OnSwitch) здійснити включення / вимикання спалаху. Виводити прискорення акселерометра в Label. При зрушуванні Switch (OnSwitch) здійснити вібрацію на мобільному пристрої. Виводити прискорення акселерометра в Label.

Контрольні питання

1. Як програмно здійснити вібрацію на мобільному пристрої?
2. Який клас використовується для доступу до вбудованого акселерометра?
3. Як створити обробник події OnTimer?
4. Як налаштувати інтервал таймера?

5 ЛАБОРАТОРНА РОБОТА №5

Тема. Використання намірів (intents) при програмуванні на мобільні пристрої.

Мета. На практиці вивчити призначення андроїд намірів для відкриття різних файлів, здійснення дзвінків з мобільного пристрою, відкриття будь-якого URL адресу в браузері.

Теоретичні відомості

Intent є механізм опису однієї операції, яку необхідно виконати - вибрати зображення, відправити повідомлення, відправити лист і т.д. Багато додатків використовують Intent для своєї роботи. Найбільш часто використовується можливість Intent для запуску в своєму додатку іншої активності (наприклад, викликати зі свого додатка Dropbox для збереження файлу, або запустити іншу активність) [15,16,17].

Тип різних дій

ACTION_VIEW – перегляд даних;

ACTION_SEND – відправка даних;

ACTION_SENDTO - відправлення повідомлень для окремого контакту, вказаного в URI;

ACTION_CALL - формувати звернення по телефону;

ACTION_WEB_SEARCH - відкрити активність для пошуку в інтернеті, ґрунтуючись на тексті, переданому за допомогою шляху URI.

У даній лабораторній роботі для використання намірів застосовуються Java класи.

Для роботи з SD картою існують спеціальні функції, які відображають повний шлях до стандартних папок на карті пам'яті (рінгтони, картинки і т.д.).

Наприклад `TPath.GetSharedPicturesPath` відображає шлях / storage / sdcard0 / Pictures стандартної папки з зображеннями.

Нижче показані стандартні шляхи, які можна використовувати для зберігання файлів і їх пошуку (рис. 5.1) [18].

Функції	Deployment Manager	На вашому пристрої	Доступ
GetHomePath	.\assets\internal\	'/data/data/<application ID>/files'	Закрит
TPath.GetHomePath	.\assets\internal\	'/data/data/<application ID>/files'	Закрит
TPath.GetTempPath	.\assets\temp\	'/storage/emulated/0/Android/data/<application ID>/files/tmp'	Открит
TPath.GetLibraryPath	-	'/data/app-lib/<application ID>-1'	Закрит
TPath.GetDocumentsPath	.\assets\internal\	'/data/data/<application ID>/files'	Закрит
TPath.GetSharedDocumentsPath	.\assets\	'/storage/emulated/0/Android/data/<application ID>/files'	Открит
TPath.GetCachePath	-	'/data/data/<application ID>/cache'	Закрит
TPath.GetPicturesPath	.\assets\Pictures\	'/storage/emulated/0/Android/data/<application ID>/files/Pictures'	Открит
TPath.GetSharedPicturesPath	-	'/storage/emulated/0/Pictures'	Открит
TPath.GetPublicPath	.\assets\	'/storage/emulated/0/Android/data/<application ID>/files'	Открит
TPath.GetCameraPath	.\assets\DCIM\	'/storage/emulated/0/Android/data/<application ID>/files/DCIM'	Открит
TPath.GetSharedCameraPath	-	'/storage/emulated/0/DCIM'	Открит
TPath.GetMusicPath	.\assets\Music\	'/storage/emulated/0/Android/data/<application ID>/files/Music'	Открит
TPath.GetSharedMusicPath	-	'/storage/emulated/0/Music'	Открит
TPath.GetMoviesPath	.\assets\Movies\	'/storage/emulated/0/Android/data/<application ID>/files/Movies'	Открит
TPath.GetSharedMoviesPath	-	'/storage/emulated/0/Movies'	Открит
TPath.GetAlarmsPath	.\assets\Alarms\	'/storage/emulated/0/Android/data/<application ID>/files/Alarms'	Открит
TPath.GetSharedAlarmsPath	-	'/storage/emulated/0/Alarms'	Открит
TPath.GetRingtonesPath	.\assets\Ringtones\	'/storage/emulated/0/Android/data/<application ID>/files/Ringtones'	Открит
TPath.GetSharedRingTonesPath	-	'/storage/emulated/0/Ringtones'	Открит
TPath.GetDownloadsPath	.\assets\Download\	'/storage/emulated/0/Android/data/<application ID>/files/Download'	Открит
TPath.GetSharedDownloadsPath	-	'/storage/emulated/0/Download'	Открит

Рисунок 5.1 – Стандартні системні шляхи для збереження та завантаження файлів

Як приклад використання намірів буде завантажено файл зображення “mypicture.jpg”, яке лежить в папці /mnt / sdcard / mypicture / jpg

```

procedure TForm2.Button2Click(Sender: TObject);
var Intent: JIntent;
begin
    Intent := TJIntent.Create;
    Intent.setAction(TJIntent.JavaClass.ACTION_VIEW);
    Intent.setDataAndType(StrToJURI('file:' + '/mnt / sdcard / mypicture.jpg'),
StringToJString('image / jpg'));
    SharedActivity.startActivity(Intent);
end;

```

Розглянемо більш детально наступний рядок програмного коду:

```

Intent.setDataAndType(StrToJURI('file:' + '/mnt/sdcard/' + ListBox1.Selected.Text),
StringToJString('image / jpg'));

```

У цьому рядку вказується:

- повний шлях до файлу, при цьому потрібно додати 'file:', інакше операційна система Android не “зрозуміє”, що потрібно відкрити файл;
- другим параметром йде вказівка MIME Type.

Також можна вказувати неповний MIME тип, наприклад, так “image / *”, “video / *” і т.п. Типів популярних медіа файлів наведена у табл. 5.1.

Таблиця 5.1 – Типи популярних медіафайлів

Тип файлів	Розширення	MIME Type
Android Application	.apk	application / vnd.android.package-archive
Text	.txt	text / plain
	.csv	text / csv
	.xml	text / xml
Web related	.htm	text / html
	.html	text / html
	.php	text / php
Image	.png	image / png
	.gif	image / gif
	.jpg	image / jpg
	.jpeg	image / jpeg
	.bmp	image / bmp
Audio	.mp3	audio / mp3
	.wav	audio / wav
	.ogg	audio / x-ogg
	.mid	audio / mid
	.midi	audio / midi
	.amr	audio / AMR
Video	.mpeg	video / mpeg
	.3gp	video / 3gpp
Package	.jar	application / java-archive
	.zip	application / zip
	.rar	application / x-rar-compressed
	.gz	application / gzip

У даній лабораторній роботі використаний компонент ScrollBox, який дозволяє розширити простір активності (форми). Також він дозволяє прокрутити вміст активності, яке не вміщується на екрані мобільного пристрою. Інтерфейс розробленого демонстраційного мобільного додатку лабораторної роботи має вигляд (рис. 5.2).

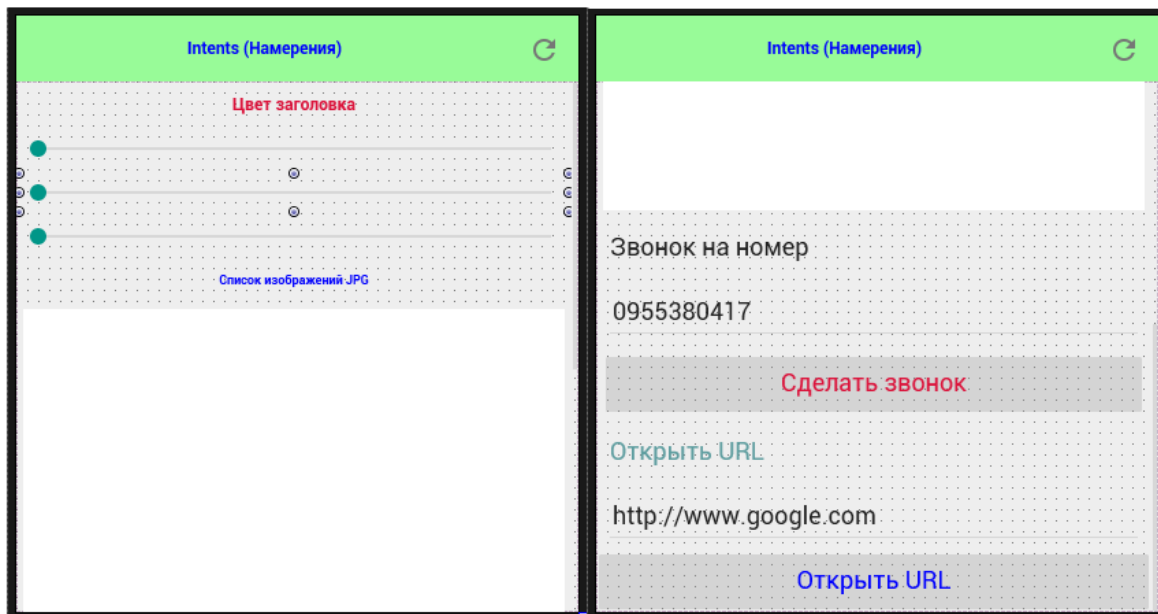


Рисунок 5.2 – Інтерфейс розробленого мобільного додатку

Лістинг програми

```

unit Main;
interface
uses
  System.SysUtils, System.Types, System.UITypes, System.Classes, System.Variants,
  FMX.Types, FMX.Controls, FMX.Forms, FMX.Graphics, FMX.Dialogs, FMX.Layouts,
  FMX.ListBox, FMX.StdCtrls, FMX.Controls.Presentation, System.IOUtils,
  FMX.Helpers.Android, Androidapi.Helpers,
  Androidapi.JNI.GraphicsContentViewText, FMX.Edit, FMX.Gestures;
type
  TIntentForm = class (TForm)
    ToolBar1: TToolBar;
    Button1: TButton;
    Label1: TLabel;
    ListBox1: TListBox;
    Button2: TButton;
    Label2: TLabel;
    TrackBar1: TTrackBar;
    TrackBar2: TTrackBar;
    TrackBar3: TTrackBar;
    Label3: TLabel;
    ScrollBox1: TScrollBox;
    Label4: TLabel;
    Edit1: TEdit;
    GestureManager1: TGestureManager;
    Label5: TLabel;
    Edit2: TEdit;
    Button3: TButton;
    procedure Button1Click (Sender: TObject);
    procedure TrackBar1Change (Sender: TObject);
    procedure ListBox1Gesture (Sender: TObject);
  end;

```

```

    const EventInfo: TGestureEventInfo; var Handled: Boolean);
    procedure Button2Click (Sender: TObject);
    procedure Button3Click (Sender: TObject);
private
    {Private declarations}
public
    {Public declarations}
end;
var
    IntentForm: TIntentForm;
implementation
{$ R * .fmx}
// Функція зміни кольору по каналах RGB
Function ARGB (const A, R, G, B: byte): TAlphaColor;
begin
    Result := B + G SHL 8 + R SHL 16+ A SHL 24;
end;
// Пошук файлів і додавання їх в ListBox
procedure TIntentForm.Button1Click (Sender: TObject);
var f: TSearchRec;
begin
    // Очищення списку
    ListBox1.Items.Clear;
    // Пошук першого файлу
    // if FindFirst ( '/ mnt / sdcard / *. Jpg', faAnyFile, f) <> 0 then exit;
    // Висновок шляху / Pictures
    Label3.Text := TPath.GetSharedPicturesPath;
    if FindFirst (TPath.GetSharedPicturesPath + '/ *. jpg', faAnyFile, f) <> 0 then exit;
    Listbox1.Items.Add (f.name); // Додаємо знайдений файл в список
    while FindNext (f) = 0 do // шукаємо інші файли
    begin
        Listbox1.Items.Add (f.name);
    end;
    FindClose (f); // Закриваємо файлову змінну
end;
// Зателефонувати за номером
procedure TIntentForm.Button2Click (Sender: TObject);
var Intent: JIntent;
begin
    Intent := TJIntent.Create;
    Intent.setAction (TJIntent.JavaClass.ACTION_CALL);
    Intent.setData (StrToJUri ( 'tel:' + edit1.Text));
    SharedActivity.startActivity (Intent);
end;
// Відкрити URL у браузері
procedure TIntentForm.Button3Click (Sender: TObject);
var Intent: JIntent;
begin
    Intent := TJIntent.Create;

```

```

Intent.setAction (TJIntent.JavaClass.ACTION_VIEW);
Intent.setData (StrToJURI (Edit2.Text));
SharedActivity.startActivity (Intent);
end;
// Відкриття зображень через Інтеніт
procedure TIntentForm.ListBox1Gesture (Sender: TObject;
const EventInfo: TGestureEventInfo; var Handled: Boolean);
var
Intent: JIntent;
begin
// Для виклику програми для перегляду зображення в списку ListBox
// потрібно натиснути і утримувати елемент списку ListBox з ім'ям зображення
case EventInfo.GestureID of
// Довге натиснення
igilongTap:
// Викликаємо намір для відкриття зображення
begin
Intent := TJIntent.Create;
Intent.setAction (TJIntent.JavaClass.ACTION_VIEW);
//Intent.setDataAndType(StrToJURI('file: '+' /mnt/sdcard/'+ListBox1.Selected.Text),
StringToJString ( 'image / jpg'));
Intent.setDataAndType (StrToJURI ( 'file:' + TPath.GetSharedPicturesPath + '/' +
ListBox1.Selected.Text), StringToJString ( 'image / jpg'));
SharedActivity.startActivity (Intent);
end; end; end;
// Міняємо колір ToolBar
procedure TIntentForm.TrackBar1Change (Sender: TObject);
begin
ToolBar1.TintColor := ARGB (255, Round (Trackbar1.Value), Round (Trackbar2.Value),
Round (Trackbar3.Value));
end; end.

```

Індивідуальне завдання

Наведений приклад програмного коду мобільного додатка для демонстрації роботи з намірами необхідно скопіювати. Далі встановити арк на мобільному пристрої. Розібратися з використанням намірів в мобільних додатках. Змінити тип відкриваються медіафайлів (MIME TYPE) для пошуку і відкриття через Інтеніт. Змінити шляху пошуку медіафайлів (використовуючи TPath). Для зберігання номерів використовувати ComboEdit. Передбачити збереження списку номерів в список (використовуючи ComboEdit) для подальшого виклику через механізм намірів.

Контрольні запитання

1. Що являють собою андроїд наміри? Наведіть приклади використання.
2. Яким чином можна здійснити виклик за допомогою Intent?

6 ЛАБОРАТОРНА РОБОТА №6

Тема. Ознайомлення з MultiTouch.

Мета. Ознайомитися з технологією MultiTouch на конкретному прикладі програми, що використовує дану технологію.

Теоретичні відомості

Мультитач (від англ. Multi-touch - «множинний дотик») - функція сенсорних систем введення (сенсорний екран, сенсорна панель), що здійснює одночасне визначення координат двох і більше точок торкання. Мультитач використовується у жестових інтерфейсів наприклад для зміни масштабу зображення. При збільшенні відстані між точками дотику відбувається збільшення зображення. Крім того, мультитач-екрани дозволяють працювати з пристроєм одночасно декільком користувачам.

Мультитач дозволяє не тільки визначити взаємне розташування декількох точок дотику в кожен момент часу, але і визначити пару координат для кожної точки дотику, незалежно від їх положення відносно один одного та кордонів сенсорної панелі. Правильне розпізнавання всіх точок дотику збільшує можливість інтерфейсу сенсорної системи введення. Коло розв'язуваних завдань при використанні функції мультитач залежить від швидкості, ефективності та інтуїтивності її застосування.

Хід роботи

Для демонстрації роботи мультитач використовується подія головною активності (форми) OnTouch. При торкання до екрану будуть перегляньте окружності в місцях торкання і відповідно написи з координатами торкань.

Лістинг програми

```
unit Main;
interface
uses
System.SysUtils, System.Types, System.UITypes, System.Classes, System.Variants,
    FMX.Types, FMX.Controls, FMX.Forms, FMX.Graphics, FMX.Dialogs, FMX.StdCtrls,
    FMX.Layouts, FMX.Controls.Presentation, FMX.Objects;
type
TForm2 = class (TForm)
    procedure FormTouch (Sender: TObject; const Touches: TTouches;
        const Action: TTouchAction);
private
    {Private declarations}
    FCountTouch: Integer;
public
    {Public declarations}
```

```

end;
var
  Form2: TForm2;
implementation
{$ R * .fmx}
uses
  System.TypInfo;
procedure TForm2.FormTouch (Sender: TObject; const Touches: TTouches;
  const Action: TTouchAction);
var
  i, j: Integer;
  MyNewCircle: TCircle; // Динамічні окружності
  LabelNew: TLabel; // Динамічні написи
begin
  // Опитуємо кількість дотиків
  if FCountTouch <> Length (Touches) then
    begin
      // Видаляємо всі компоненти на формі
      Form2.DeleteChildren;
      // В залежності від числа дотиків створюємо окружності
      for i: = 1 to Length (Touches) do
        begin
          MyNewCircle: = TCircle.Create (Form2);
          MyNewCircle.Parent: = Form2;
          MyNewCircle.Name: = 'C' + i.ToString;
          MyNewCircle.Size.Height: = 40;
          MyNewCircle.Size.Width: = 40;
          // Зафарбовуємо окружності в певний колір
          // в залежності від дотику
          case i of
            1: MyNewCircle.Fill.Color: = TAlphaColors.Red;
            2: MyNewCircle.Fill.Color: = TAlphaColors.Blue;
            3: MyNewCircle.Fill.Color: = TAlphaColors.Green;
            4: MyNewCircle.Fill.Color: = TAlphaColors.Chocolate;
            5: MyNewCircle.Fill.Color: = TAlphaColors.Greenyellow;
          else
            begin
              MyNewCircle.Fill.Color: = TAlphaColors.Aqua;
            end;
          end;
        end;
      // Створюємо написи для виведення координат дотику
      LabelNew: = TLabel.Create (Form2);
      LabelNew.Parent: = Form2;
      LabelNew.Name: = 'Label' + i.ToString;
      LabelNew.AutoSize: = True;
      LabelNew.TextSettings.Trimming: = TTextTrimming.None;
      LabelNew.TextSettings.WordWrap: = False;
      LabelNew.TextSettings.FontColor: = TAlphaColors.Red;
    end;
  end;
end;

```

```

    end;
end;
for j: = 0 to Length (Touches) -1 do
begin
// Виводимо окружності в залежності від координат дотику
with Form2.FindComponent ( 'C' + IntToStr (j + 1)) as TCircle do
begin
    Position.X: = Trunc (Touches [j] .Location.X);
    Position.Y: = Trunc (Touches [j] .Location.Y);
end;
// Виводимо написи і координати торкань
with Form2.FindComponent ( 'Label' + IntToStr (j + 1)) as TLabel do
begin
    Position.X: = Trunc (Touches [j] .Location.X + 30);
    Position.Y: = Trunc (Touches [j] .Location.Y + 30);
    Text: = 'Дотик № ' + (j + 1) .ToString + '('
    + Trunc (Touches [j] .Location.X) .ToString + 'x'
    + Trunc (Touches [j] .Location.Y) .ToString + ')';
end;
end;
end;
// Підрахунок торкань
FCountTouch: = Length (Touches);
end; end.

```

Функція FindComponent дозволяє знайти об'єкт, який знаходиться на формі. Це дуже зручно, коли необхідно за назвою об'єкта знайти сам об'єкт або групу об'єктів.

Індивідуальне завдання

Скомпілювати і встановити на мобільному пристрої приклад цього додатка. Перевірити дію мультитач торкаючись до екрану.

Контрольні питання

1. Поясніть поняття “MultiTouch”? Для чого може бути використана дана технологія?
2. Поясніть призначення функції FindComponent?
3. Як динамічно створити об'єкт? Наведіть приклад у вигляді фрагмента програмного коду?

7 ЛАБОРАТОРНА РОБОТА №7

Тема. Google Maps. Їх використання у мобільних додатках.

Мета. Ознайомитися з технологією використання карт у мобільних додатках. Навчитися використовувати класи TMapView та TLocationSensor для відображення карт та геолокації. Навчитися елементарним маніпуляціям з об'єктами карт, такими як маркери та лінії.

Теоретичні відомості

Клас TMapView використовується для відображення карт у мобільному додатку. За допомогою класу TLocationSensor можливо визначити координати (широту та довготу) місцезнаходження мобільного пристрою. За допомогою маркерів можна відображувати місцезнаходження об'єктів згідно їх координат (latitude/longitude). Для коректної роботи TMapView необхідно отримати Google API Key. Для цього необхідно мати аккаунт Google.

Для реєстрації та отримання ключа Google API необхідно перейти за посиланням та натиснути Get Started (рис. 7.1).

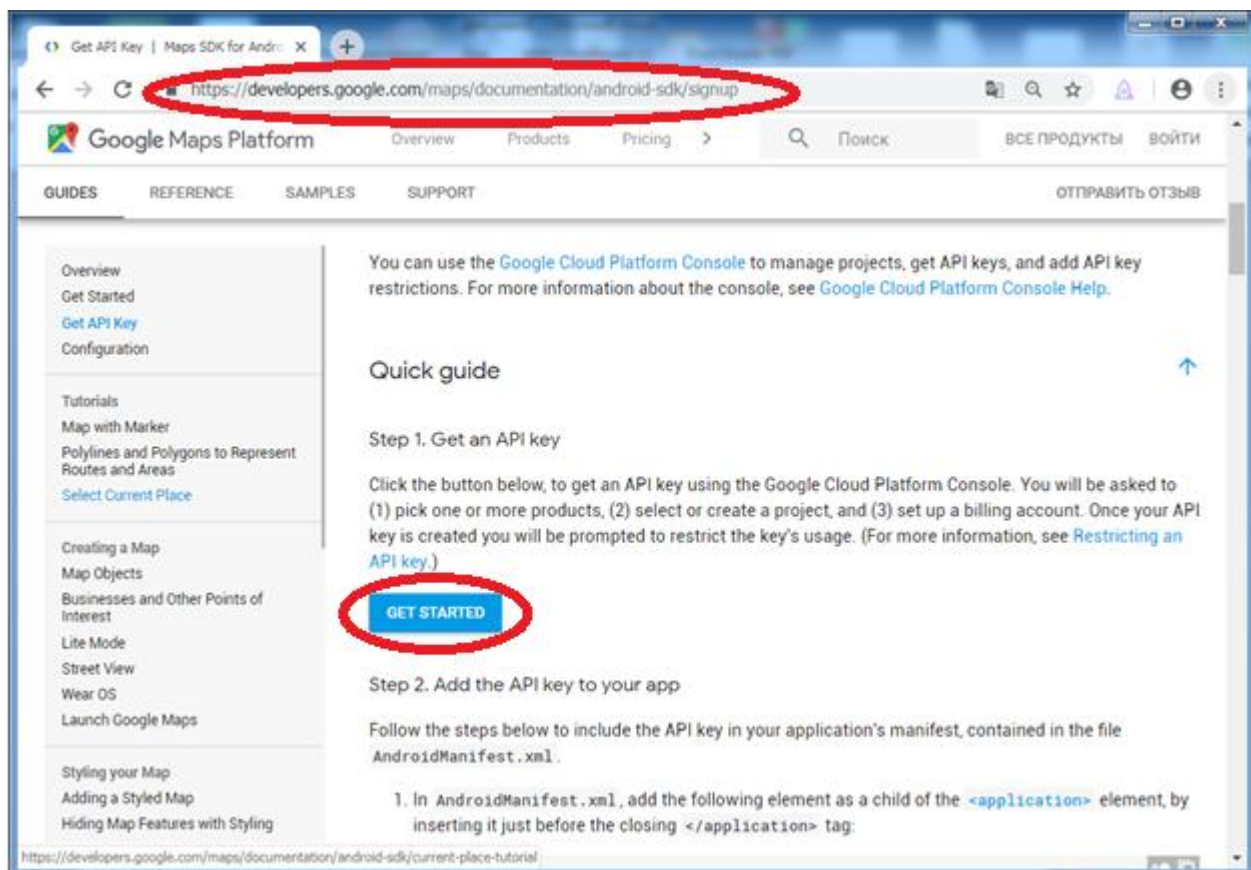


Рисунок 7.1 – Google Maps Platform

Далі необхідно вибрати API (можна вибрати все рис. 7.2).

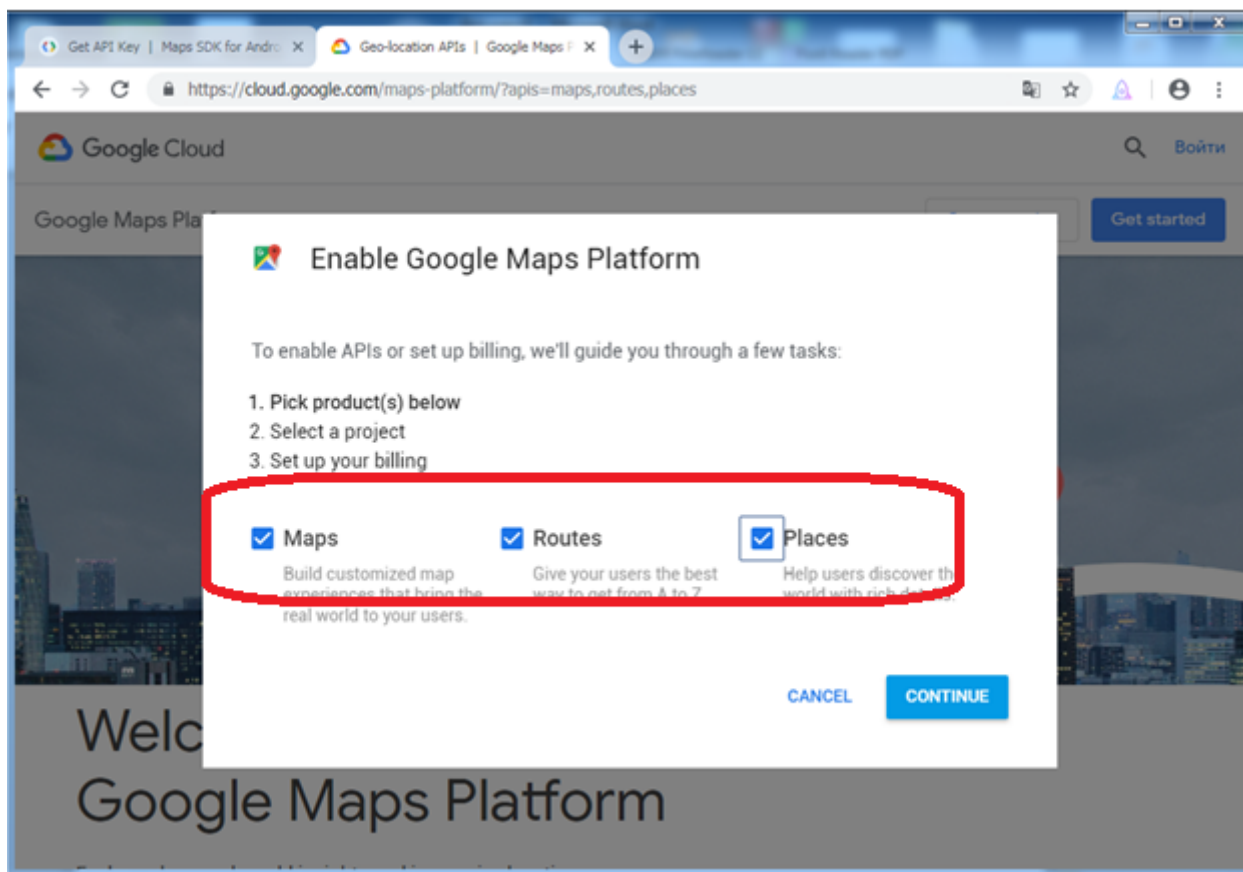


Рисунок 7.2 – Geo-location APIs

Якщо є вже створений аккаунт Google, можна скористатися ним для входу в Google API Console, якщо нема – необхідно його створити та скористатися ним для реєстрації та отримання ключа Google API Key (рис. 7.3).

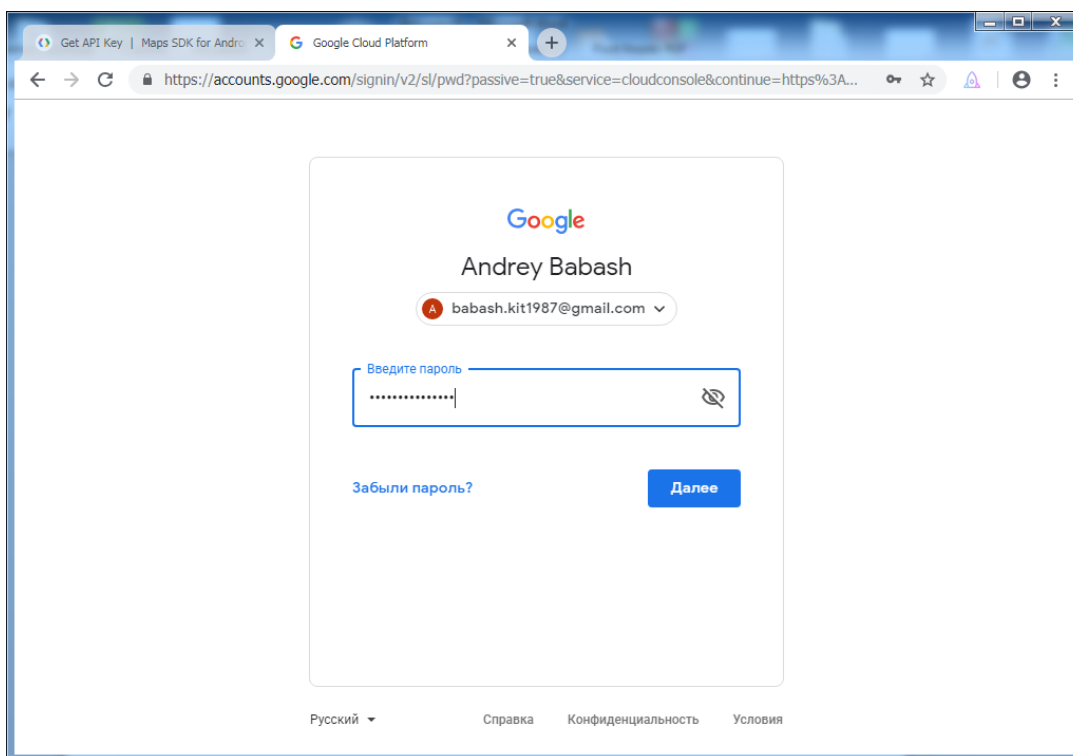


Рисунок 7.3 – Вхід в Google API console

Потім необхідно або створити, або вибрати проект (рис. 7.4). Платіжний аккаунт можна не створювати. Він необхідний для побудови маршрутів на карті.

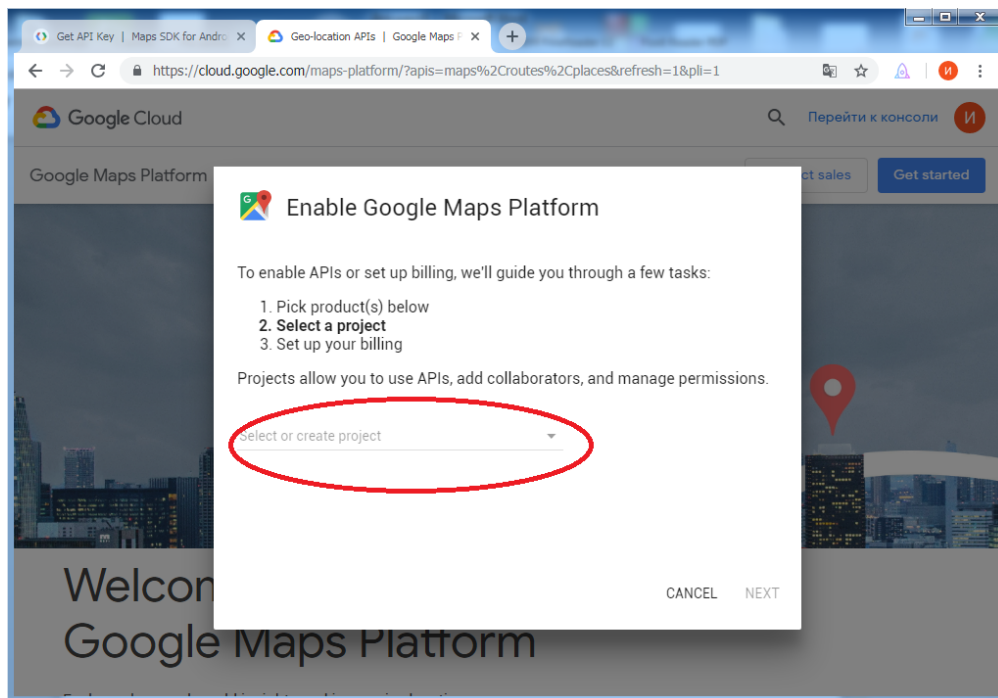


Рисунок 7.4 – Створення проекту

Для отримання ключа необхідно зайти в меню Учетные данных (рис. 7.5).

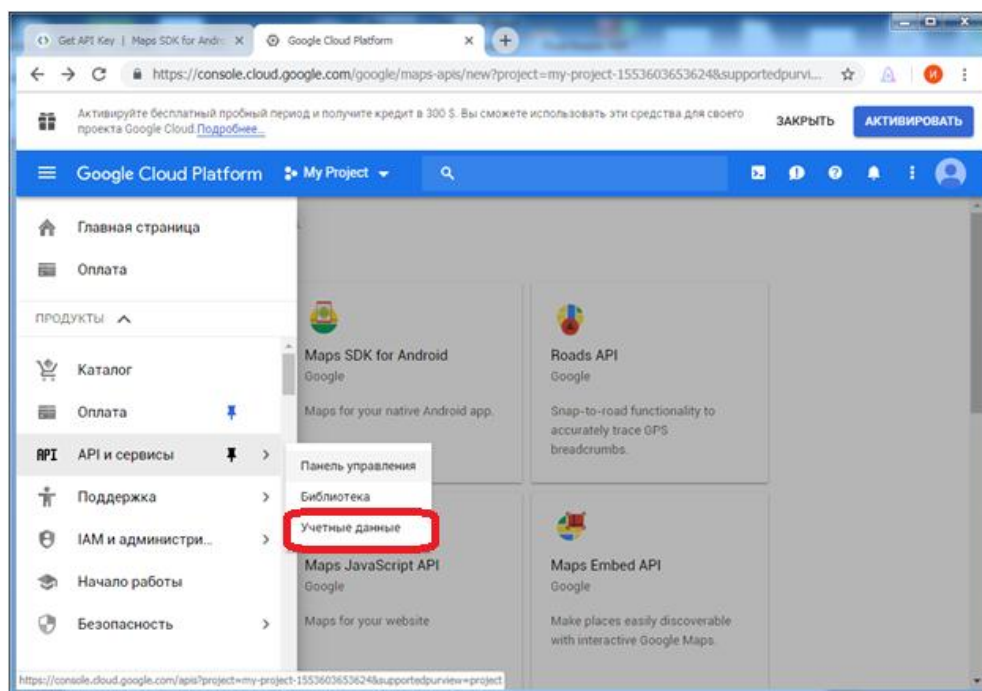


Рисунок 7.5 – Вікно з проектом у консолі Google API

Після необхідно створити облікові дані (рис. 7.6).

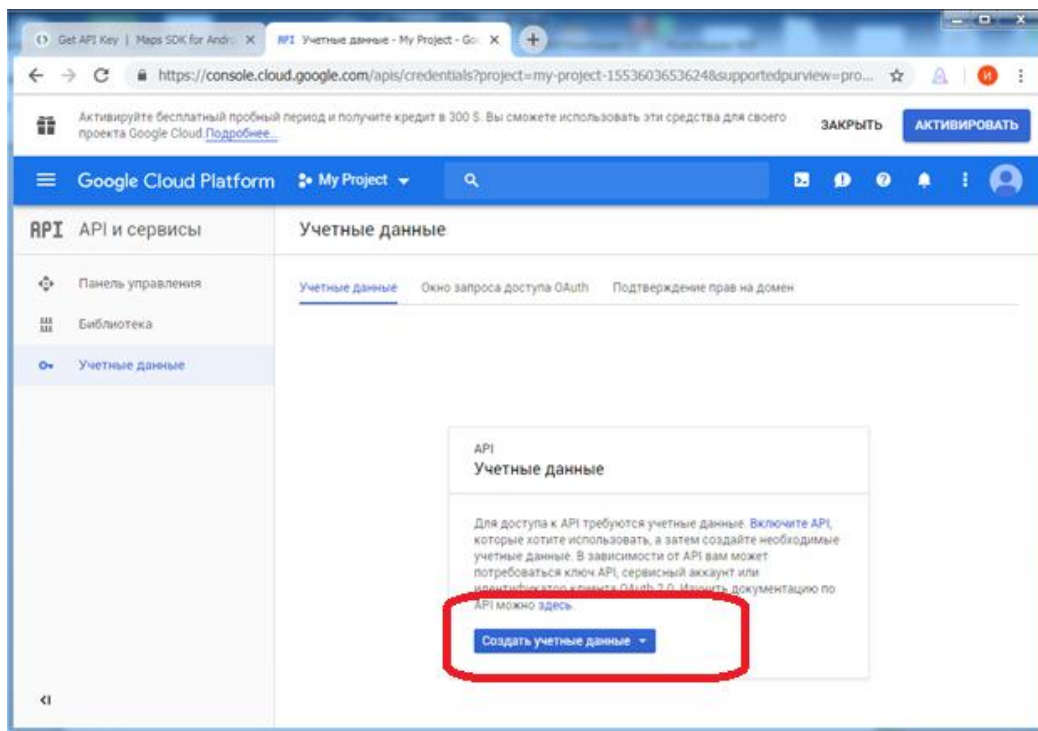


Рисунок 7.6 – Створення облікових даних

Далі вибрати Ключ API (рис. 7.7).

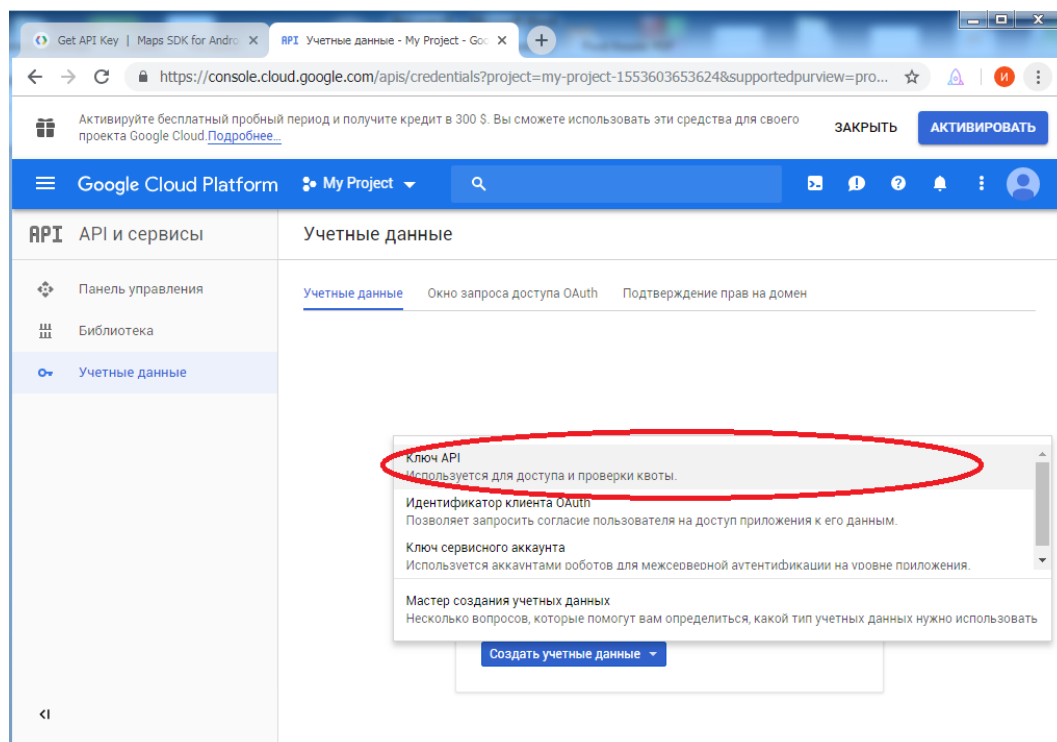


Рисунок 7.7 – Вікно вибору створення Ключа API

Тепер можна просто скопіювати ключ для його використання у проєкті (рис. 7.8)

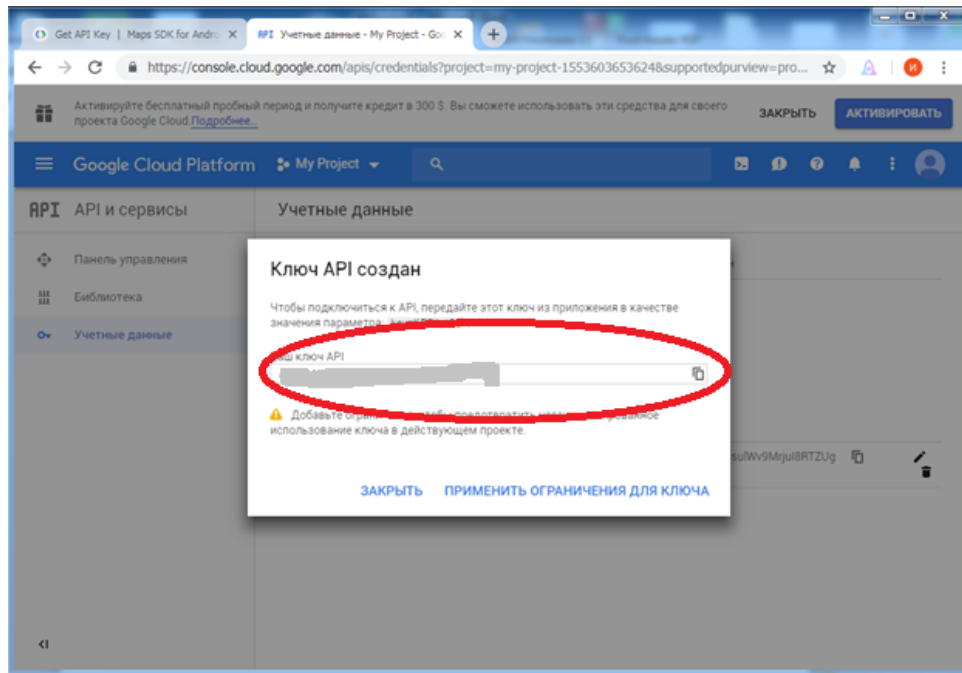


Рисунок 7.7 – Ключ Google API, необхідний для роботи з картами

Можна вибрати обмеження для використання ключа за бажанням.

Хід роботи

Для використання в проєкті отриманого Google API Key необхідно у середовищі розробки Rad Studio вибрати меню Project -> Options -> Application -> Version Info. Необхідно додати раніше отриманий ключ (рис. 7.8).

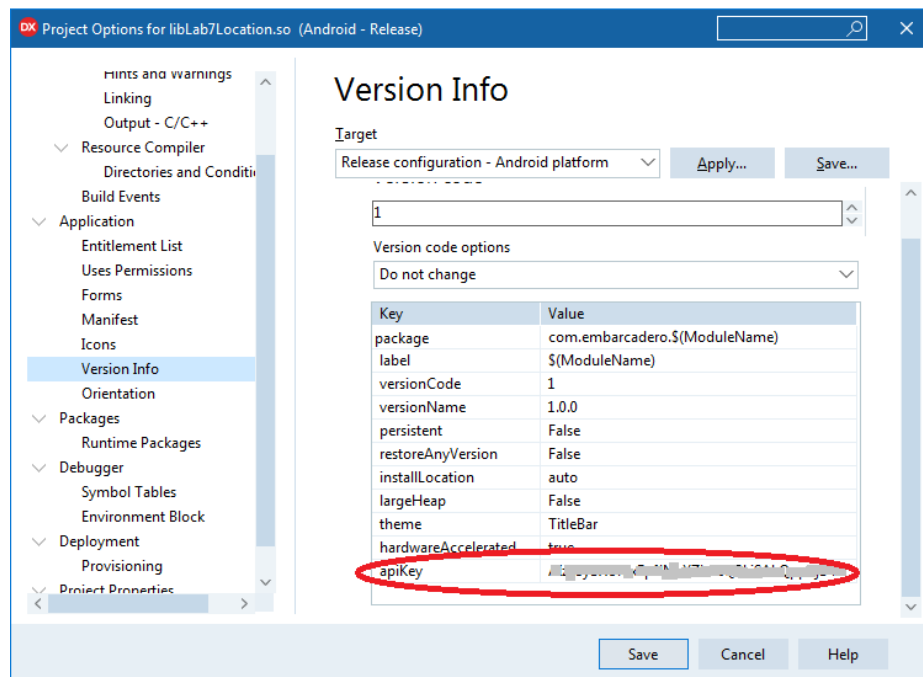


Рисунок 7.8 – Додавання ключа Google API в проєкт

Для коректної роботи Google Play Services та Google Maps необхідно у меню Entitlement List вибрати підтримку всіх сервісів та натиснути Save (рис. 7.9).

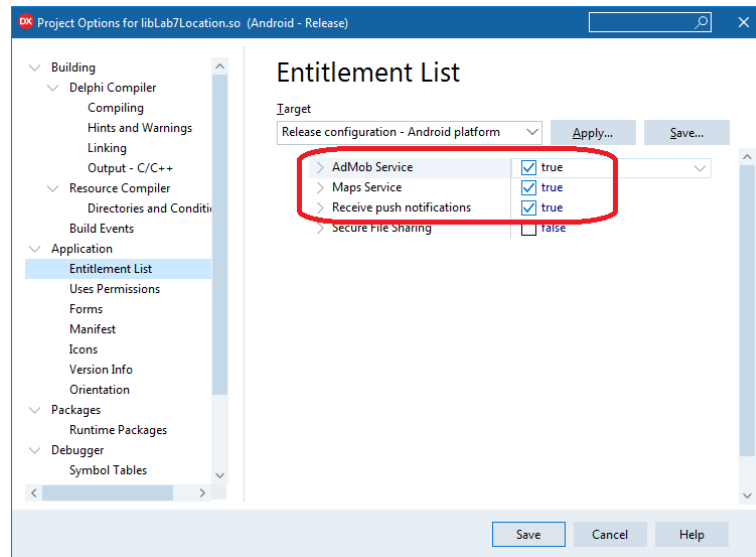


Рисунок 7.8 – Додавання підтримки сервісів у проект

У проекті представлена демонстрація використання карт у мобільних додатках. Місцезнаходження мобільного пристрою визначається за допомогою класу `TLocationSensor`, який представлений у вигляді невізуального компонента на головній формі. За допомогою його події `OnLocationChanged` можна відстежувати поточне місцезнаходження пристрою. Для його активізації необхідно властивість `Active` перевести у режим `True`. Це можна зробити у інспекторі об'єктів, або програмно (рис. 7.9).

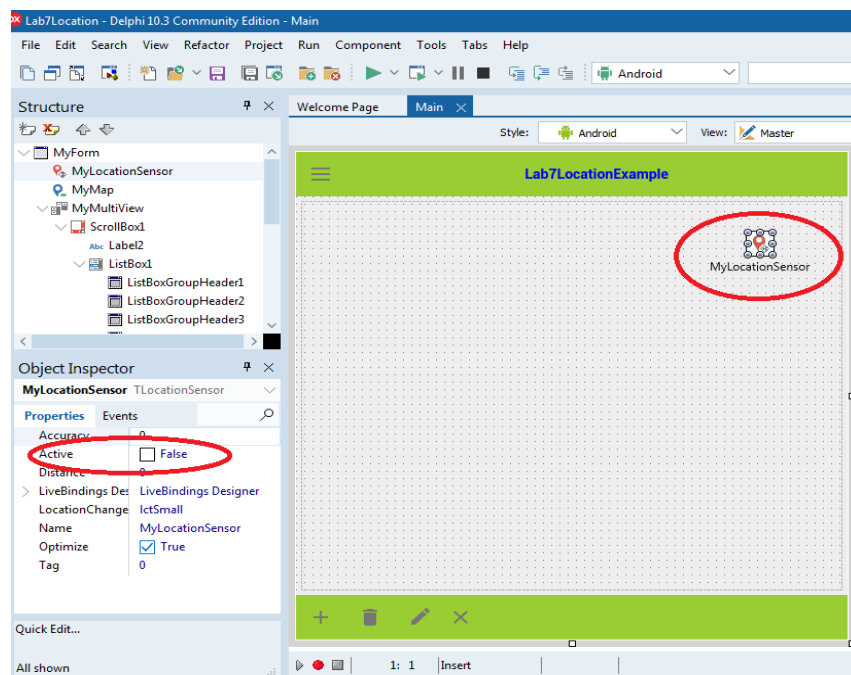


Рисунок 7.9 – Головна форма демонстраційного проекту

Також в Delphi для створення меню є дуже зручний компонент MultiView. Він може слугувати контейнером для інших компонентів, таким чином створюючи різноманітні складні інтерфейси. Керувати появою та зникненням меню можна за допомогою кнопок. Встановити потрібну кнопку можна за допомогою властивості MasterButton. За допомогою режиму Mode необхідно встановити Drawer (меню, яке виїжджає). Видимість на формі у режимі розробки можна встановлювати за допомогою властивості Visible (рис. 7.10).

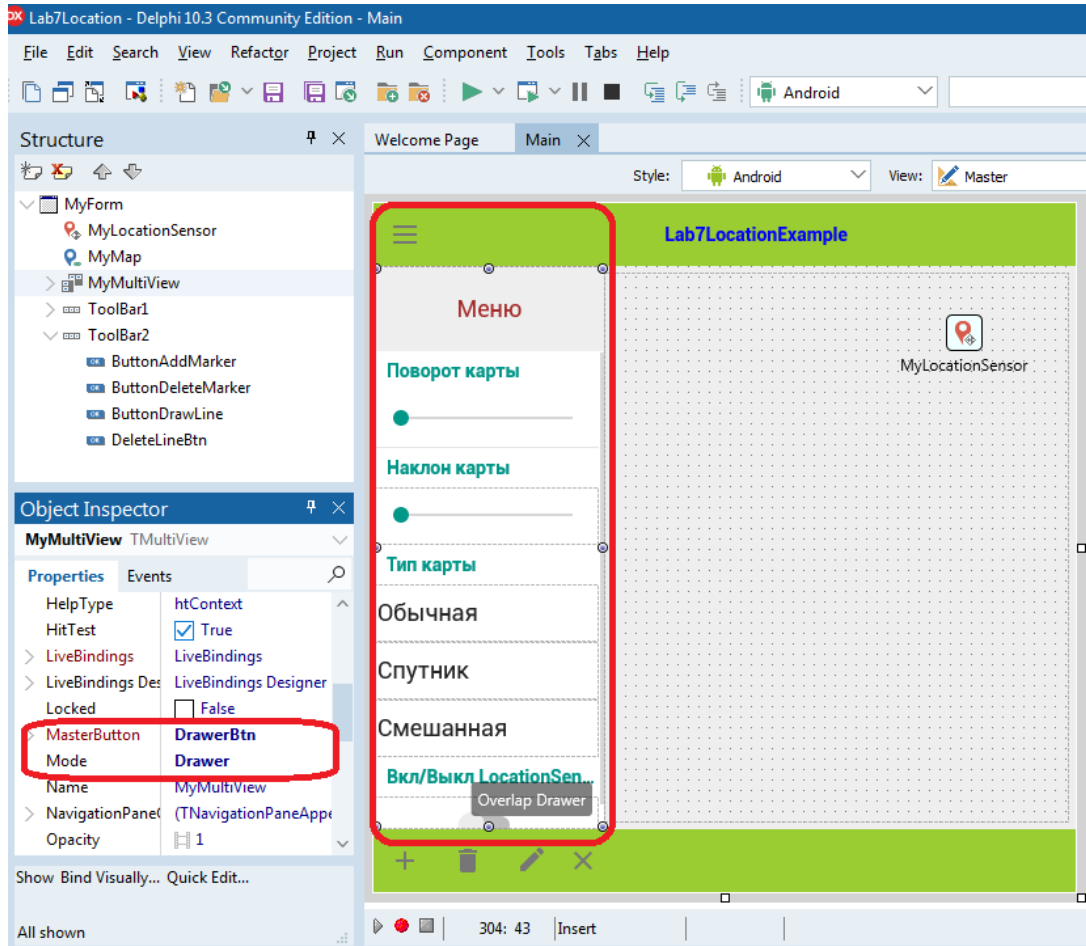


Рисунок 7.10 – Компонент MultiView

Лістинг програми

```
unit Main;

interface

uses
  System.SysUtils, System.Types, System.UITypes, System.Classes, System.Variants,
  FMX.Types, FMX.Controls, FMX.Forms, FMX.Graphics, FMX.Dialogs, FMX.Maps,
  FMX.StdCtrls, FMX.Controls.Presentation, FMX.MultiView, FMX.ListBox,
  FMX.Layouts, System.Sensors, System.Sensors.Components
  {$IFDEF ANDROID}
```

```

    , FMX.Platform.Android, Androidapi.JNI.Widget, FMX.Helpers.Android, AndroidA-
pi.Helpers
{$ENDIF};

```

```

type

```

```

TMyForm = class(TForm)
    ToolBar1: TToolBar;
    ToolBar2: TToolBar;
    DrawerBtn: TButton;
    Label1: TLabel;
    MyMap: TMapView;
    ButtonAddMarker: TButton;
    ButtonDeleteMarker: TButton;
    MyMultiView: TMultiView;
    ScrollBox1: TScrollBox;
    Label2: TLabel;
    ListBox1: TListBox;
    ListBoxGroupHeader1: TListBoxGroupHeader;
    ListBoxItem1: TListBoxItem;
    TrackBarRotate: TTrackBar;
    ListBoxGroupHeader2: TListBoxGroupHeader;
    ListBoxItem2: TListBoxItem;
    TrackBarTilt: TTrackBar;
    ListBoxGroupHeader3: TListBoxGroupHeader;
    ListBoxItem3: TListBoxItem;
    ListBoxItem4: TListBoxItem;
    ListBoxItem5: TListBoxItem;
    ListBoxGroupHeader4: TListBoxGroupHeader;
    ListBoxItem6: TListBoxItem;
    LocationSwitch: TSwitch;
    MyLocationSensor: TLocationSensor;
    ButtonDrawLine: TButton;
    DeleteLineBtn: TButton;
    procedure LocationSwitchSwitch(Sender: TObject);
    procedure FormCreate(Sender: TObject);
    procedure MyLocationSensorLocationChanged(Sender: TObject;
        const OldLocation, NewLocation: TLocationCoord2D);
    procedure MyMapMarkerClick(Marker: TMapMarker);
    procedure ButtonAddMarkerClick(Sender: TObject);
    procedure ButtonDeleteMarkerClick(Sender: TObject);
    procedure MyMultiViewStartHiding(Sender: TObject);
    procedure MyMultiViewStartShowing(Sender: TObject);
    procedure ListBoxItem3Click(Sender: TObject);
    procedure ListBoxItem4Click(Sender: TObject);
    procedure ListBoxItem5Click(Sender: TObject);
    procedure FormKeyDown(Sender: TObject; var Key: Word; var KeyChar: Char;
        Shift: TShiftState);
    procedure FormKeyUp(Sender: TObject; var Key: Word; var KeyChar: Char;
        Shift: TShiftState);

```



```

procedure TrackBarRotateChange(Sender: TObject);
procedure TrackBarTiltChange(Sender: TObject);
procedure ButtonDrawLineClick(Sender: TObject);
procedure DeleteLineBtnClick(Sender: TObject);
private
  { Private declarations }
public
  { Public declarations }
  // Маркер местоположения
  MyLocationMarker:TMapMarker;
  // Маркер пользователя
  MyMarker:TMapMarker;
  // Координаты маркеров местоположения и пользователя
  MyLocationPoint:TPointF;
  MyMarkerPoint:TPointF;
  // Описание линии
  MyLineDescriptor:TMapPolylineDescriptor;
  // Линия
  MyLine:TMapPolyline;
  // Точки координат линии
  Points:TArray<TMapCoordinate>;
end;

var
  MyForm: TMyForm;

implementation

{$R *.fmx}

// Подтверждение выхода из приложения

procedure ExitConfirm;
begin

  // Вывод диалогового окна

  MessageDlg('Хотите выйти!', System.UITypes.TMsgDlgType.mtInformation,
  [
    System.UITypes.TMsgDlgBtn.mbYes,
    System.UITypes.TMsgDlgBtn.mbNo
  ], 0,

  procedure(const AResult: TModalResult)
  begin
    case AResult
    of

```

```

mrYES:

begin

    {$IFDEF ANDROID}
        MainActivity.finish; // Выход из приложения
    {$ENDIF}

    {$IFDEF MSWINDOWS}
        Application.Terminate;
    {$ENDIF}

end;

mrNo:

begin

end;

end;
end
)

end;

// Добавляем пользовательский маркер на карту
procedure TMyForm.ButtonAddMarkerClick(Sender: TObject);
var MarkerLocation: TMapCoordinate; // координаты маркера
    MyMarkerDescr: TMapMarkerDescriptor; // описание внешнего вида маркера
begin
    // Проверяем, есть ли уже маркер на карте (проверяем на "пустоту")
    if MyMarker=nil then
        begin
            MarkerLocation := TMapCoordinate.Create(MyMarkerPoint); // Переводим
координаты из TPointF в TMapCoordinate
            MyMarkerDescr := TMapMarkerDescriptor.Create(MarkerLocation, 'Это мой
маркер'); // Создаем описание маркера
            MyMarkerDescr.Draggable := True; // Маркер можно перемещать на карте
            MyMarker := MyMap.AddMarker(MyMarkerDescr); // Присваиваем маркеру описа-
ние и добавляем на карту
        end;
    end;

// Удаляем маркер
procedure TMyForm.ButtonDeleteMarkerClick(Sender: TObject);
begin
    // Проверяем, есть ли маркер на карте

```

```

    if MyMarker<>nil then
    begin
        MyMarker.Remove; // Удаляем маркер
        MyMarker:=nil; // присваиваем маркеру "пустоту"
    end;
end;

// Рисуем линию
procedure TMyForm.ButtonDrawLineClick(Sender: TObject);
begin
    // Проверяем, есть ли уже линия на карте
    if MyLine=nil then
    begin
        // Устанавливаем размер динамического массива (для построения линии нужно
две точки)
        SetLength(Points,2);
        // Линия будет соединять маркер местоположения и пользовательский маркер
        // Создаем точки на основе координат местоположения маркеров
        Points[0] := TMapCoordinate.Create(MyLocationPoint);
        Points[1] := TMapCoordinate.Create(MyMarkerPoint);
        // Создаем описание линии на основе точек
        MyLineDescriptor := TMapPolylineDescriptor.Create(Points);
        // Устанавливаем толщину линии
        MyLineDescriptor.StrokeWidth := 20;
        // Устанавливаем цвет линии
        MyLineDescriptor.StrokeColor := TAlphaColors.YellowGreen;
        // Добавляем линию на карту
        MyLine := MyMap.AddPolyline(MyLineDescriptor);
    end
    else
    begin
        // Если линия уже добавлена - удаляем
        Myline.Remove;
        Myline := nil;
    end;
end;

// Удаление линии
procedure TMyForm.DeleteLineBtnClick(Sender: TObject);
begin
    // Если линия есть на карте - удаляем
    if MyLine<>nil then
    begin
        Myline.Remove;
        Myline := nil;
    end;
end;

procedure TMyForm.FormCreate(Sender: TObject);

```

```

begin
// Координаты пользовательского маркера (широта и долгота)
  MyMarkerPoint.X:=41.2;
  MyMarkerPoint.Y:=40.5;
end;

// При нажатии кнопки Назад запрашиваем диалог выхода из приложения
procedure TMyForm.FormKeyDown(Sender: TObject; var Key: Word; var KeyChar: Char;
  Shift: TShiftState);
begin
  if Key = vkHardwareBack then ExitConfirm;
end;

// При отпускании ничего не делаем
procedure TMyForm.FormKeyUp(Sender: TObject; var Key: Word; var KeyChar: Char;
  Shift: TShiftState);
begin
  if Key = vkHardwareBack then Key := 0;
end;

// Устанавливаем тип карты
procedure TMyForm.ListBoxItem3Click(Sender: TObject);
begin
  MyMap.MapType := TMapType.Normal;
end;

procedure TMyForm.ListBoxItem4Click(Sender: TObject);
begin
  MyMap.MapType := TMapType.Satellite;
end;

procedure TMyForm.ListBoxItem5Click(Sender: TObject);
begin
  MyMap.MapType := TMapType.Hybrid;
end;

//*****//
// При переключении свича активируем или деактивируем LocationSensor
procedure TMyForm.LocationSwitchSwitch(Sender: TObject);
begin
  if LocationSwitch.IsChecked then
    begin
      MyLocationSensor.Active:=true;
      {$IFDEF ANDROID}
        TJToast.JavaClass.makeText(TAndroidHelper.Context, StrToJCharSe-
quence('Location Sensor On'), TJToast.JavaClass.LENGTH_LONG).show;
      {$ENDIF}
    end
  else

```

```

begin
    MyLocationSensor.Active:=false;
    {$IFDEF ANDROID}
        TJToast.JavaClass.makeText(TAndroidHelper.Context, StrToJCharSe-
quence('Location Sensor Off'), TJToast.JavaClass.LENGTH_LONG).show;
    {$ENDIF}
end;
end;

// Обработчик события изменения местоположения
procedure TMyForm.MyLocationSensorLocationChanged(Sender: TObject;
    const OldLocation, NewLocation: TLocationCoord2D);
var MyMarkerLocationDescr: TMapMarkerDescriptor; MyLocation: TMapCoordinate;
begin
    // Перерисовываем карту
    MyMap.Repaint;
    // Присваиваем координаты нового местоположения
    MyLocationPoint.X := NewLocation.Latitude;
    MyLocationPoint.Y := NewLocation.Longitude;

    // Проверяем, есть ли на карте маркер текущего местоположения
    if MyLocationMarker=nil then
        begin
            // Координаты местоположения (вычисляется посредством GPS датчика или Wi-Fi
сетей)
            MyLocation := TMapCoordinate.Create(MyLocationPoint);
            // Перемещаем карту согласно координатам местоположения
            MyMap.Location := MyLocation;
            // Создаем описание маркера
            MyMarkerLocationDescr := TMapMarkerDescriptor.Create(MyLocation,'Я
здесь!));');
            // Настраиваем внешний вид маркера
            with MyMarkerLocationDescr do
                begin
                    Draggable := False; // запрет на перемещение по карте
                    Visible := True; // видимость маркера
                    Appearance := TMarkerAppearance.Billboard; // внешний вид - объемный мар-
кер
                    Snippet := Format('Lat/Lon:
%s,%s',[FloatToStrF(MyLocationPoint.X,ffGeneral,4,2),FloatToStrF(MyLocationPoint.Y,ffGeneral,
4,2)]); // описание под названием
                end;

            MyLocationMarker := MyMap.AddMarker(MyMarkerLocationDescr); // Добавляем
маркер на карту
            MyMap.Zoom:=30; // зум карты 30
        end
    else
        // Если маркер уже существует на карте - удаляем

```

```

begin
    MyLocationMarker.Remove;
    MyLocationMarker := nil;
end;
end;

// Обработчик кликов по маркерам
procedure TMyForm.MyMapMarkerClick(Marker: TMapMarker);
var MarkerTitle:string;
begin
    // запрашиваем название
    MarkerTitle:=Marker.Descriptor.Title;
    // в зависимости от названия выводим "тост"
    {$IFDEF ANDROID}
    if MarkerTitle<>'Я здесь!))'; then
        TJToast.JavaClass.makeText(TAndroidHelper.Context, StrToJCharSe-
quence('Координаты: '+Marker.Descriptor.Position.Latitude.ToString+'
'+Marker.Descriptor.Position.Longitude.ToString), TJToast.JavaClass.LENGTH_LONG).show
    else
        TJToast.JavaClass.makeText(TAndroidHelper.Context, StrToJCharSequence('Мой
маркер!))'), TJToast.JavaClass.LENGTH_LONG).show;
    {$ENDIF}
end;

// При скрытии MultiView показываем карту
procedure TMyForm.MyMultiViewStartHiding(Sender: TObject);
begin
    if not MyMap.Visible then MyMap.Visible:=true;
end;

// При появлении MultiView скрываем карту
procedure TMyForm.MyMultiViewStartShowing(Sender: TObject);
begin
    if MyMap.Visible then MyMap.Visible:=not MyMap.Visible;
end;

// Поворот карты
procedure TMyForm.TrackBarRotateChange(Sender: TObject);
begin
    MyMap.Bearing := TrackBarRotate.Value;
end;

// Наклон карты
procedure TMyForm.TrackBarTiltChange(Sender: TObject);
begin
    MyMap.Tilt := TrackBarTilt.Value;
end;

end.

```

Для коректного визначення місцеположення необхідно дозволити на пристрої визначення місцеположення та у середовищі розробки для проекту встановити Permissions (Access Fine Location, Access Coarse Location – приблизне та точне місцеположення).

Індивідуальне завдання

Використовуючи поля вводу TEdit реалізувати введення координат користувальницького маркера у Run Time режимі, тобто щоб користувач міг змінювати координати з мобільного додатку.

Контрольні питання

1. Поясніть призначення Google API Key?
2. Як додати маркер до карти?
3. Як додати або видалити лінію?

ВИСНОВКИ

У рамках методичного посібника було розглянуто спосіб розробки сучасних нативних мобільних додатків операційної системи Android з використанням крос-платформного середовища Embarcadero Delphi. Розглянуті приклади розробки стандартних інтерфейсів мобільних систем. Наведений приклад роботи з Ini файлами та їх використання при збереженні та завантаженні індивідуальних користувальницьких налаштувань.

Наведені приклади використання графіки на малювання на екрані мобільного пристрою з використанням класу TCanvas. Для обробки та використання жестів у мобільних додатках використовується клас TGestureManager, який дозволяє легко використовувати та обробляти жести.

Також представлений приклад та методика роботи з текстовими файлами на мобільному пристрої з використанням Object Pascal.

Наміри (intents) використовуються для здійснення дозвону на певний номер, для завантаження веб ресурсу у браузер, для завантаження зображень та ін.

Розглянуто використання класу TTimer для відліку певного часу. Представлений приклад використання вібрації з використанням Java класів. Наведено використання вбудованого акселерометра мобільного пристрою, та його вбудованої камери.

Таким чином, середовище Embarcadero Delphi дозволяє створювати нативні додатки для різних популярних операційних систем з використанням єдиної кодової бази. Також воно дозволяє розширення, тобто можна створити та встановити будь-який унікальний клас, встановити його та використовувати у мобільній розробці. Середовище дозволяє використовувати нативні класи операційної системи Android. Для цього створені спеціальні модулі – обгортки рідних Java класів.

Середовище є гнучким та дозволяє швидко та ефективно у короткий час розробити бажаний додаток для замовника.

Недоліками середовищами є те, що кінцевий файл має відносно великий розмір, так як включає всі бібліотеки FireMonkey. Деякі мобільні пристрої не підтримуються середовищем.

ПЕРЕЛІК ПОСИЛАНЬ

1. Осипов, Д.Л. Delphi. Программирование для Windows, IOS и Android / Осипов Д.Л. // СПб.: БХВ-Петербург, 2014. – 464 с.: ил.
2. Н. Культин Основы программирования в Embarcadero Delphi / Культин Н.Б. 2015, – 232 с.
3. Культин, Н. Б. Delphi в задачах и примерах / Культин Н. Б. – 2-е изд., перераб. и доп. – СПб. : БХВ-Петербург, 2008. — 288 с. : ил.
4. Дарахвелидае, П. Г. Программирование в Delphi 7 / Дарахвелидае П. Г., Марков Е. П. – СПб. : БХВ-Петербург, 2003. – 784 с : ил. – ISBN 5-94157-116-X.
5. Delphi 7 / под общ. ред. А.Д. Хомоненко. – СПб. : БХВ-Петербург, 2008. – 1216 с. : ил.
6. В. В. Леонов Обучение мобильной разработке на Delphi / Леонов В.В. // Проектное обучение – 342 с.
7. Шкрыль А. А. Delphi. Народные советы / Шкрыль А. А. - СПб. : БХВ-Петербург, 2007. - 400 с. : ил. – ISBN 978 -5-9775-0047-0.
8. Кэнту М. Delphi 7 для профессионалов / Кэнту М. – СПб. : Питер, 2004. – 1101 с. : ил. ISBN 5-94723-593-5.
9. Фленов М. Е. Программирование в Delphi глазами хакера / Фленов М. Е. – СПб. : БХВ-Петербург, 2003. – 368 с. : ил. ISBN 5-94157-351-0.
10. Marco Cantu Object Pascal Handbook / Marco Cantu // *Piacenza, Italy*, 2016 – pp. 563, ISBN-10: 1514349949
11. Фелкер, Донн Android. Разработка приложений для чайников. : Пер. с англ. — М. : ООО “И.Д. Вильямс”, 2012. — 336 с. : ил.
12. Дэрсси Л. Android за 24 часа. Программирование приложений под операционную систему Google/ Дэрсси Л., КондерШ. — М.: Рид Групп, 2011. — 464 с.
13. Я. Файн Программирование на Java для детей, родителей, дедушек и бабушек / Я. Файн – 231 с.
14. Программирование в Delphi на Андроид. Библиотека Firemonkey. [Электронный ресурс] Режим доступа: URL: https://ridero.ru/books/delphi_programmirovanie_dlya_android
15. FireMonkey [Электронный ресурс] Режим доступа: <http://firemonkey.ru>
16. Android Context [Электронный ресурс] Режим доступа: <http://www.fandroid.info/context-kontekst-v-android-chto-eto-kak-poluchit-i-zachem-ispolzovat/>
17. Intents в Delphi [Электронный ресурс] Режим доступа: <http://progerson.ru/2644-delphi-firemonkey-intent-2-zvonok-i-sms-soobscheniya.html>
18. Delphi XE5-Berlin. Разработка под Андроид [Электронный ресурс] Режим доступа: <http://www.webdelphi.ru/2014/01/delphi-xe5-ispolzovanie-intent-namereniya-v-android/>