**Projekt Grupowy - Google Glass  
Dokumentacja Aplikacji testowej**

**Leszek Buława**

**Karol Kawalerczyk**

**Maria Kucułyma  
 Tomasz Scharmach**

Contents

[Informacje ogólne 2](#_Toc423538377)

[Wykorzystywane technologie 2](#_Toc423538378)

[Glass Development Kit (API 19) 2](#_Toc423538379)

[Działanie aplikacji 2](#_Toc423538380)

[Opis działania 2](#_Toc423538381)

[Schemat blokowy aplikacji 3](#_Toc423538382)

[Implementacja 3](#_Toc423538383)

[Wstęp 3](#_Toc423538384)

[Klasa GetFonts 4](#_Toc423538385)

[Klasa MainActivity 5](#_Toc423538386)

[Rozpoznawanie mowy 6](#_Toc423538387)

[Diagram klas 7](#_Toc423538388)

[Zdjęcia aplikacji 7](#_Toc423538389)

# Informacje ogólne

Na potrzeby projektu przeprowadzone zostały badania dzięki którym możliwe było określenie parametrów takich jak czcionka, rozmiar tekstu i ilość wyrazów widoczna na okularach. W tym celu stworzona została aplikacja testowa napisana w języku Java Android wkorzystująca GDK(Glass Development Kit).

Aplikacja składa się z 4 kart. Na każdej z kart widoczny jest fragment tekstu *Lorem ipsum* który pozwala skoncentrować uwagę na aspektach wizualnych. Aplikacja wyświetla tekst w czterech czcionkach: Arial, Calibri, Impact oraz Times New Roman. Użytkownik w menu aplikacji ma możliwość zmiany rozmiaru czcionki oraz ilości wyświetlanego tekstu na ekranie okularów.

# Wykorzystywane technologie

* Eclipse Luna x64
* ObjectAid Plugin
* Glass Development Kit (API 19)
* Github
* Android SDK
* Android Screen Monitor
* Google Glass XE22

# Glass Development Kit (API 19)

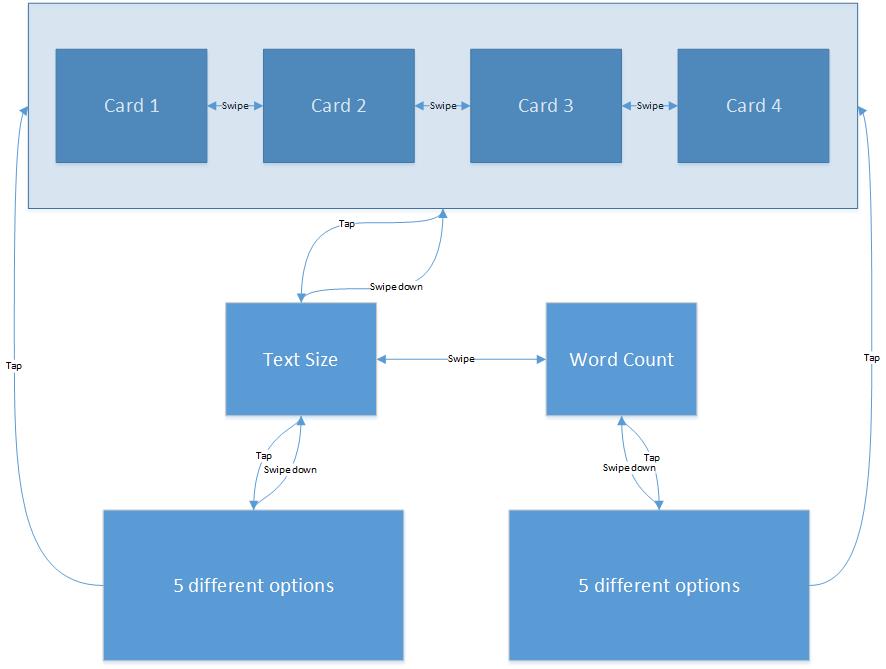
Wraz z okularami, powstało nowe API dedykowane specjalnie do nich, opierające się na standardowym Android API 19. GDK zostało wzbogacone o nowe klasy dedykowane dla okularów. Przykładem jest CardBuilder. Zaletą korzystania tylko z Android API jest to, że jest bardzo szczegółowo opisana. W razie problemów można znaleźć pomoc bez większego problemu. Korzystanie z GDK jest o tyle dobre że, posiada gotowe implementacje graficzne, dedykowane tylko dla okularów (np. animacja przechodzenia między kartami). Dodatkowo, w GDK bardzo łatwo implementuje się metody pozwalające na sterowanie głosowe aplikacji.

# Działanie aplikacji

## Opis działania

Aplikację uruchamiamy z menu głównego okularów. W celu uruchomienia menu, należy wywołać polecenie głosowe „Ok Glass”, lub też naciskając panel dotykowy z boku okularów. Po otworzeniu menu głównego, należy wypowiedzieć „Start Test” lub przyciskiem na bocznym panelu wybrać aplikację. Jak wpomniano na początku, aplikacja składa się z 4 kart na których widoczny jest fragment tekstu Lorem Ipsum. Użytkownik przesuwając palcem na bocznym panelu w przód i w tył zmienia karty wyświetlane na ekranie. Kolejne karty różnią się rodzajem czionki.   
  
Naciśniecie na panel boczny powoduje otworzenie menu aplikacji w której użytkownik może wybrać dwie opcje, zmiana rozmiaru czcionki oraz ilość tekstu wyświetlanego na ekranie okularów. Użytkownik ma możliwość wyboru czionek w rozmiarze: 14, 16, 18, 20, 22 oraz ilości wyrazów na ekranie: 5-10, 10-20, 20-30, 30-40, 40-50. Po zmienieniu opcji aplikacja powraca do kart głównych ze zmienionymi wartościami. Domyślnie wybrane jest 5-10 wyrazów i czcionka o rozmiarze 14. Aby wyjść z aplikacji należy przesunąć palcem na bocznym panelu w dół.

## Schemat blokowy aplikacji



# Implementacja

## Wstęp

Rozdział ten opisuje sposoby implementacji najważniejszych elementów aplikacji. Cały projekt dostępny jest pod adresem: <https://github.com/tomcio0808/GoogleGlass.git>

Aplikacja została stworzona z wykorzystaniem Glass Development Kit działający na Androidzie 4.4.2, z wykorzystaniem środowiska Eclipse. Google Glass umożliwia programowanie aplikacji z wykorzystaniem Mirror API, jednakże rozwiązanie to umożliwia tworzenie tylko prostych programów na okulary.

Zaprojektowana aplikacja składa się z dwóch klas publicznych:

* GetFonts
* MainActivity

i jednej prywatnej:

* CustomCardScrollAdapter

## Klasa GetFonts

Google Glass umożliwia wykorzystanie wyłącznie 3 rodzajów czcionek: Roboto Light, Roboto Regular oraz Roboto Thin. Czcionki te niestety, nie różnią sie zbyt bardzo wizualnie. Z tego powodu wymagane było zaimplementowanie klasy odpowiedzialnej za importowanie innych czionek. Odpowiedzialna jest za to klasa GetFonts.

Wszystkie dostępne w aplikacji czcionki, ładowane są w konstruktorze tej klasy. Wykorzystano do tego obiektów typu Typeface. Folder z czionkami znajduje się w folderze assets projektu. Ich pobieranie odbywa się z wykorzystaniem metody createFromAsset(). Czionki importowane są w formacie ttf.

1 **public** GetFonts(Context context) {

2 **this**.context = context;

3 arialFont = Typeface.*createFromAsset*(context.getAssets(), "fonts/arial.ttf");

4 calibriFont = Typeface.*createFromAsset*(context.getAssets(), "fonts/calibri.ttf");

5 impactFont = Typeface.*createFromAsset*(context.getAssets(), "fonts/impact.ttf");

6 timesNewRomanFont = Typeface.*createFromAsset*(context.getAssets(), "fonts/timeroman.ttf");  
7 }

Klasa GetFonts posiada również dwie metody:

* getFontByID()
* getFontNameById()

Pierwsza metoda zwraca obiekt typu Typeface. Jako parametr metoda przyjmuje wartość integer. W zależności od jego wartości, z wykorzystaniem instrukcji switch zwracany jest obiekt Typeface zawierający interesującą nas czcionkę.

**public** String getFontNameByID(**int** id) {

String fontNameToReturn;

**switch** (id) {

**case** 0:

fontNameToReturn = *ARIAL*;

**break**;

**case** 1:

fontNameToReturn = *CALIBRI*;

**break**;

**case** 2:

fontNameToReturn = *IMPACT*;

**break**;

**case** 3:

fontNameToReturn = *TIMES*;

**break**;

**default**:

fontNameToReturn = *ARIAL*;

**break**;

}

**return** fontNameToReturn;

}

Metoda getFontNameById() została skonstruowana analogicznie do powyższej. Jedyną różnicą jest to, że zwraca obiekt typu String. Metoda ta wykorzystywana jest do wyświetlania nazwy aktualnie widocznej czcionki na ekrawanie okularów.

## Klasa MainActivity

Klasa MainActivity jest główną klasą aplikacji dziedziczącą po klasie Activity. Głównym widokiem aplikacji jest obiekt typu CardScrollView. Jego adapterem jest obiekt prywatnej klasy CustomCardScrollAdapter która dziedziczy po CardScrollAdapter.

1 **protected** **void** onCreate(Bundle savedInstanceState) {

2 **super**.onCreate(savedInstanceState);

3 // android.os.Debug.waitForDebugger(); //uncomment if you want to debbug in eclipse

4 getFonts = **new** GetFonts(getApplicationContext());

5 textToView = **this**.getString(R.string.***sentence\_1***);

6 createCards();

7 mCardScrollView = **new** CardScrollView(**this**);

8 mAdapter = **new** CustomCardScrollAdapter();

9 mCardScrollView.setAdapter(mAdapter);

10 mCardScrollView.activate();

11

12 setContentView(mCardScrollView);

13 }

Głównym elementem aplikacji są obiekty typu CardBuilder. Klasa CardBuilder została wprowadzona w GDK zastępując dotychczasową klasę Card i na jej obiektach są wyświetlane elementy widoczne na ekranie okularów. Aplikacja posiada 4 karty, po jednej dla każdej z czcionek. Ich tworzenie odbywa się w metodzie createCards()

1 **private** **void** createCards() {

2 mCards = **new** ArrayList<CardBuilder>();

3 mCards.add(**new** CardBuilder(**this**, CardBuilder.Layout.***EMBED\_INSIDE***).setEmbeddedLayout(R.layout.***activity\_main***));

4 mCards.add(**new** CardBuilder(**this**, CardBuilder.Layout.***EMBED\_INSIDE***).setEmbeddedLayout(R.layout.***activity\_main***));

5 mCards.add(**new** CardBuilder(**this**, CardBuilder.Layout.***EMBED\_INSIDE***).setEmbeddedLayout(R.layout.***activity\_main***));

6 mCards.add(**new** CardBuilder(**this**, CardBuilder.Layout.***EMBED\_INSIDE***).setEmbeddedLayout(R.layout.***activity\_main***));

7 }

Metoda setEmbeddedLayout() oznacza, że tworzona karta będzie miała layout zdefiniowany w pliku xml. Layout ten składa się z jednego elementu TextView rozciągniętego na cały obszar ekranu okularów.

Wyświetlanie kart odbywa się za pomocą metody getView() znajdującej się w klasie CustomCardScrollAdapter dziedziczącej po CardScrollAdapter. W metodzie znajduje się instrukcja switch, która pozwala na zmianę kart.

1 **public** View getView(**int** position, View convertView, ViewGroup parent) {

2 View view = **new** CardBuilder(getApplicationContext(), CardBuilder.Layout.***EMBED\_INSIDE***)   
3 .setEmbeddedLayout(R.layout.***activity\_main***.getView();

4 String currentHeaderText;

5 String currentFontName;

6 Typeface currentFont;

7

8 bodyText = (TextView) view.findViewById(id.***bodyTextView***);

9

10 **switch** (position) {

11 **case** 0: {

12 currentFontName = getFonts.getFontNameByID(position);

13 currentFont = getFonts.getFontByID(position);

14 **break**;

15 }

16 **case** 1: {

17 currentFontName = getFonts.getFontNameByID(position);

18 currentFont = getFonts.getFontByID(position);

19 **break**;

20 }

21 **case** 2: {

22 currentFontName = getFonts.getFontNameByID(position);

23 currentFont = getFonts.getFontByID(position);

24 **break**;

25 }

26 **case** 3: {

27 currentFontName = getFonts.getFontNameByID(position);

28 currentFont = getFonts.getFontByID(position);

29 **break**;

30 }

31 **default**:;

32 currentFontName = getFonts.getFontNameByID(0);

33 currentFont = getFonts.getFontByID(0);

34 **break**;

35 }

36

37 currentHeaderText = "Czcionka " + currentFontName + ", rozmiar " + textSize + ",

38

39 bodyText.setText(currentHeaderText+"\n"+textToView);

40 bodyText.setTypeface(currentFont);

41 bodyText.setTextColor(Color.***WHITE***);

42 bodyText.setTextSize(TypedValue.***COMPLEX\_UNIT\_DIP***, textSize);

43

44 **return** view;

45 }

Menu tworzone jest za pomocą występującej standardowo w systemie android metody onCreateOptionsMenu. Jego składniki tworzone są za pomocą metody onOptionsItemSelected zawierającej prostą instrukcję switch. Każda z opcji wywołuje metodę adaptera informującą o zmianie zestawu danych (notifyDataSetChanged). Menu otwierane jest po naciśnięciu panelu dotykowego – uruchomienie Key Event (KEYCODE\_DPAD\_CENTER).

## Rozpoznawanie mowy

Aplikacja może zostać uruchomiona za pomocą polecenia głosowego. Został stworzony nowy plik, voice\_trigger.xml w katalogu res/xml. Zdefiniowano w nim trigger:

1<!--?xml version="1.0" encoding="utf-8"?-->

2

3 <trigger keyword=*"@string/glass\_startApp\_trigger"* />

4

Co ma owy trigger zrobić, zostało zdefiniowane w pliku manifest aplikacji:

19 <activity

20 android:name=*"com.projektgrupowy.MainActivity"*

21 android:label=*"@string/app\_name"* >

22 <intent-filter>

23 <action android:name=*"com.google.android.glass.action.VOICE\_TRIGGER"* />

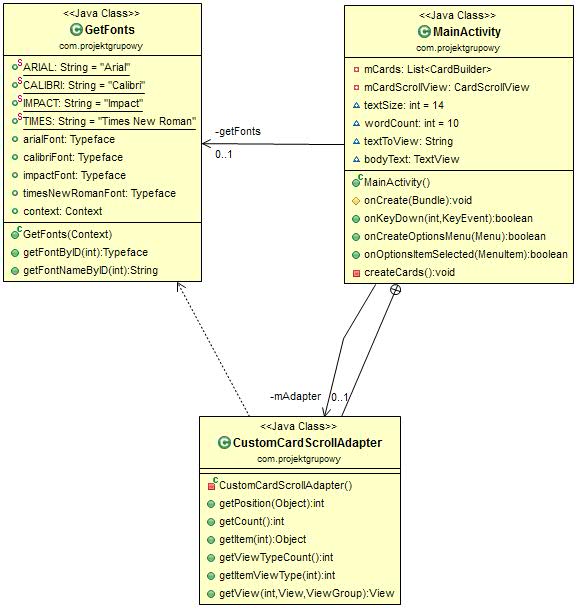
24 </intent-filter>

25 <action android:name=*"com.google.android.glass.action.VOICE\_TRIGGER"* />

26 <meta-data android:name=*"com.google.android.glass.VoiceTrigger"* android:resource=*"@xml/voice\_trigger"* />

27 </activity>

# Diagram klas



# Zdjęcia aplikacji