Vol. 33, No. 02, pp. 672-693/Diciembre 2020

ISSN-E 1995-9516
Universidad Nacional de Ingeniería
COPYRIGHT © (UNI). TODOS LOS DERECHOS RESERVADOS
http://revistas.uni.edu.ni/index.php/Nexo
https://doi.org/10.5377/nexo.v33i02.10801

Development of software for an information system that points the way using augmented reality technology

Desarrollo de software para un sistema de información que señala el camino utilizando tecnología de realidad aumentada

Alexey I. Martyshkin[1], Elena G. Bershadskaya[2]

1. Candidate of technical sciences, docent, associate Professor of sub-department «Computers and systems», *Penza state technological University (440039, Russia, Penza, Baydukov Proyezd / Gagarin Street, 1a/11, e-mail:)*, orcid id: 0000-0002-3358-4394
2. Candidate of technical sciences, professor, professor of sub-department «Computers and systems», *Penza state technological University (440039, Russia, Penza, Baydukov Proyezd / Gagarin Street, 1a/11*

Corresponding author email: alexey314@yandex.ru

(recibido/received: 16-July-2020; aceptado/accepted: 19-September-2020)

ABSTRACT

This paper describes the development process for the map directions information system application utilizing augmented reality technology. The first part of this paper discusses issues related to the research objective formulation. The objective and tasks to be solved are determined. Theoretical information and developments in the subject area are discussed. The second part describes the developed application architecture, the choice of a programming language is explained. The third part expressively addresses the development of algorithms and the application itself. The next section contains instructions for end users. And finally, a summary of the concluded work is given and relevant conclusions are made about the obtained results. It is emphasized that the work leads to the creation of an application for route finding and navigation in augmented reality mode. It is noted that this application can be extended and enhanced by developing versions for other mobile operating systems (for example, iOS and Windows Phone).
Keywords: augmented reality, maps, mobile applications, navigation, Android, Java

RESUMEN

Este documento describe el proceso de desarrollo de la aplicación del sistema de información de direcciones de mapas que utiliza tecnología de realidad aumentada. La primera parte de este artículo analiza cuestiones relacionadas con la formulación de los objetivos de la investigación. Se determina el objetivo y las tareas a resolver. Se discuten la información teórica y los desarrollos en el área temática. La segunda parte describe la arquitectura de la aplicación desarrollada, se explica la elección de un lenguaje de programación. La tercera parte aborda

expresivamente el desarrollo de algoritmos y la propia aplicación. La siguiente sección contiene instrucciones para los usuarios finales. Y finalmente, se entrega un resumen del trabajo concluido y se extraen conclusiones relevantes sobre los resultados obtenidos. Se destaca que el trabajo lleva a la creación de una aplicación de búsqueda de rutas y navegación en modo de realidad aumentada. Cabe señalar que esta aplicación se puede ampliar y mejorar desarrollando versiones para otros sistemas operativos móviles (por ejemplo, iOS y Windows Phone).
Palabras clave: realidad aumentada, mapas, aplicaciones móviles, navegación, Android, Java

1. INTRODUCTION

Mobile devices are becoming more popular with each day, which is not surprising, as nowadays these devices help finding any necessary information or performing complex operations without much effort. The most important factors making smartphones more popular than PCs and tablets are their portability and relatively low cost.
The objective was to develop a high-quality handy application with elements of augmented reality, providing users the capabilities to create convenient routes.
At the moment, one of the most promising areas of IT development is augmented reality — an environment where the real world is augmented with digital data, perceived as real-world elements (Tsvetkov, 2017; Yakovlev & Pustov,; Yakovlev & Pustov, 2013).
When the user deviates from the route, the application helps him to find the direction towards the selected location using a direction pointer in augmented reality (AR). Therefore, this scientific paper is aimed at developing the program code for AR Navigator mobile application, which main advantages are state-of-art technologies and modern-day relevance.

Theoretical information

The developer of a mobile application is faced with the task of creating a route to a destination and incorporating elements of augmented reality. The application should have the following features:

- destination object selection;
- creation of a path to the destination object from the current coordinates;
- splitting the path into coordinates and mapping different types of pointers to them;
- drawing pointers in augmented reality.

System requirements: smartphone running Android 4.3 operating system (OS) or later, having a camera, GPS, compass, accelerometer, Internet connection; Android Studio 3.1 development environment, Java 8 programming language.
Android Studio development environment is cross-platform, allowing to develop Android applications on the following operating systems:

- Microsoft Windows 10 (32- or 64-bit);
- Mac OS X 10.10 or later;
- Ubuntu 14.04 LTS (32- or 64-bit) or later.

The input data for the application is data obtained via an Internet connection and smartphone sensors, namely: GPS, orientation sensor, compass accelerometer. In the absence of an accelerometer, the application can receive data from other sensors, such as gyroscope or magnetic field sensor. The choice of input data source is based on the availability of higher priority sources. If there's no access to a source or data available, the system automatically requests data from a lower priority source. The information is provided to users in the form of graphic elements

such as labels, arrows, shapes, etc. The data in the displayed items depends on the set of actions performed by the user.
The developed mobile application is intended to display the current location and create a route to the location selected on the map, using direction pointers in augmented reality. The choice of input data source is based on the availability of higher priority sources; if there's no access to a source or data available, the system automatically requests data from a lower priority source. The information is provided to users in the form of graphic elements such as labels, arrows, shapes, etc. The data in the displayed items depends on the set of actions performed by the user.
The developed mobile application is intended to display the current location and create a route to the location selected on the map, using direction pointers in augmented reality.
At the moment, Android OS is the most popular OS in the world (Figure 1) (https://gs.statcounter.com).

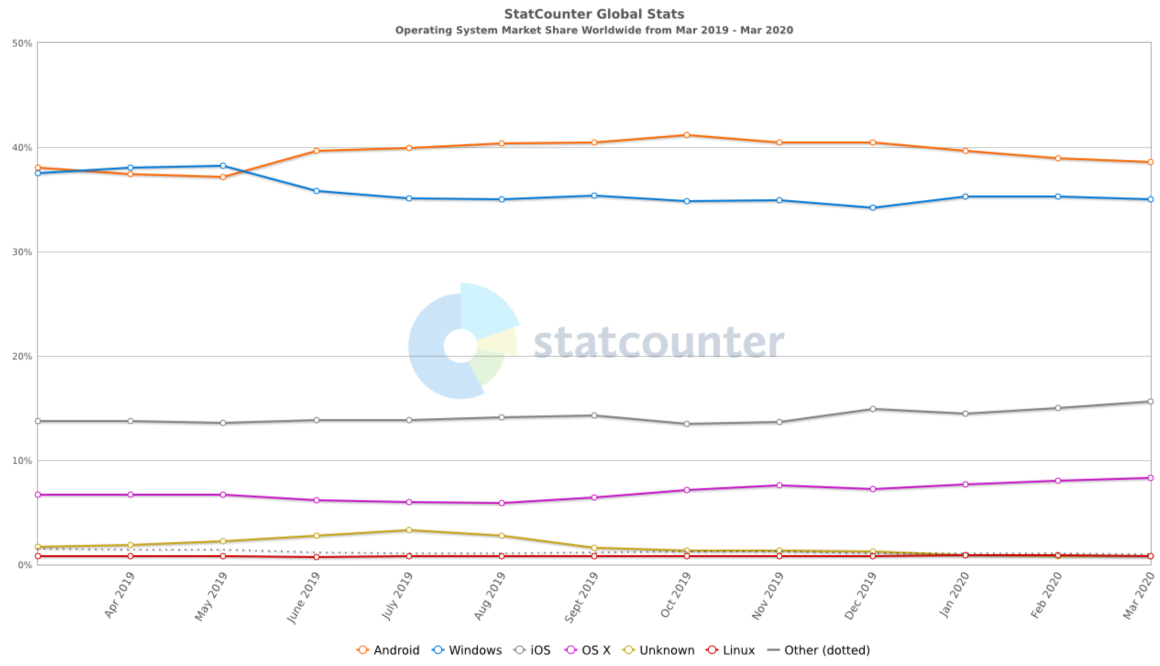


Figure 1: Operating systems popularity

Such popularity arises from Android OS simplicity, functionality, portability and supported device types. According to the official Android OS website, currently, Android OS works on such devices as smartphones, smart-watches, tablets, smart-TVs, and is also successfully integrated into cars (https://www.android.com).
Many books, documents, articles and internet resources are devoted to Android OS application development. All the necessary sources can be found in both English and Russian. Literature sources (Caudell & Mizell, 1992; Milgram & Kishino, 1994; Azuma, 1997; Boychenko & Lezhankin, 2010; Ivanova, 2018) and (Yegorov, 2019; Kuraev, 2019; Chirkin & Pimonov, 2019; Honcharova, 2019; Pavlova et al., 2020) related to the subject of this paper have been reviewed.
The book by Goloshchapov A.L. (2011) addresses the following aspects of application development: application development for mobile devices running Android OS, basic information about the Android platform, software necessary for the development of Android applications. Practical course by Reto Mayer (2011) is devoted to software development based on Android SDK 2.0 (software

development kit). The examples presented in the book reflect real programming tasks based on the studied theoretical material. Android for Programmers book (Deitel et al., 2012) contains massive amount of theory. The text of each subsection contains illustrative code samples of sixteen widely used real-world Android applications.
The official Android website for developers has a dedicated resource (http://developer.android.com) with various programming courses, API guides (API — application programming interface, a set of predefined classes, procedures, functions, structures and constants provided by the application for use in external software products (https://en.wikipedia.org)), quick references, tools for developers and application code samples.
In 2014, Google provided its vision of Android OS software and application design philosophy with the Material Design framework (https://ru.wikipedia.org). Material Design implies that applications open and collapse like cards with shadows effects (https://design.google.com).
The official JetBrains website has a dedicated resource providing useful plugins for Android Studio (https://plugins.jetbrains.com), in particular, for building UML diagrams.
One of the main unique aspects of this paper is utilization of the mobile device's integral sensors. Alexander Klimov's website (http://developer.alexanderklimov.ru) contains information about all sensor types used in mobile devices, as well as examples of their use.
Google Maps Android API is used to enable the map view, calculation of distances and areas, as well as finding directions using spherical geometry for the application. More details on this interface can be found on the official Google Maps API website (https://developers.google.com; https://developers.google.com), containing the necessary tutorials on this subject.
To implement the augmented reality feature during the application development, it is necessary to solve a number of geodesic problems, requiring understanding of the geodetic theory, in particular, a concept of azimuth (https://en.wikipedia.org).
For the initial stage of the mobile application development, the paper considers certain issues, among which, for example, is the choice of application design pattern. Existing design patterns and their implementation methods are described in the books "Design Patterns: Object-Oriented Design Techniques." (Gamma et al., 2016) and "Elemental Design Patterns" (McC Smith, 2013).
Today, the mobile application market is one of the most highly-demanded. According to the We Are Social analytical company statistics, as of April 2020, smartphones remain the main tool for accessing the Internet, and generate more web traffic than all other devices combined (Figure 2) (https://wearesocial.com).

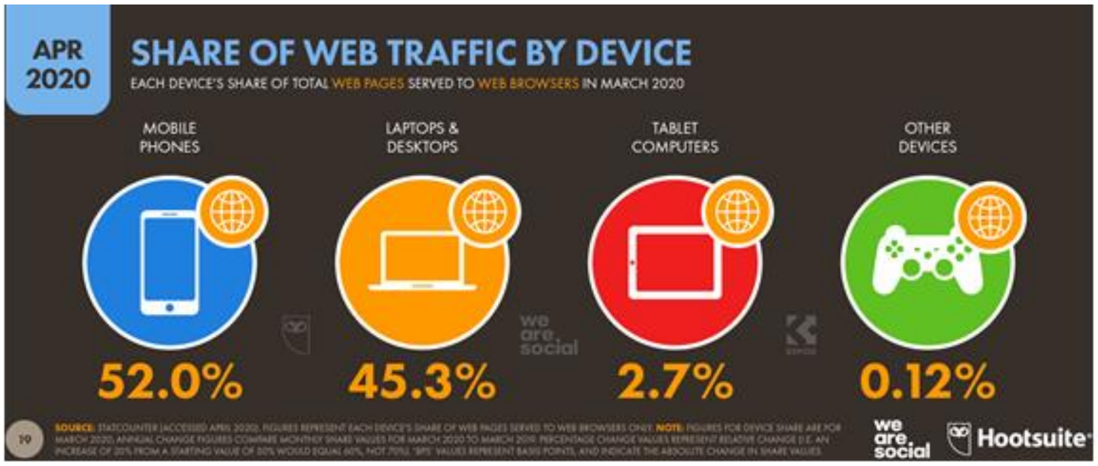


Figure 2: We Are Social statistics of online access from various devices

Android OS is the most popular OS in the world, which implies a high market demand on applications for this platform. Also, application development for this OS has a low entry threshold, as it does not require knowledge of domain-specific programming languages, and has low requirements for the computer's hardware and OS, where the development environment will be installed. For example, application development for iOS mobile OS, as well as MacOS X for Apple computers, requires knowledge of Swift programming language, and application development for Windows Phone requires the developer to have knowledge of C# and Windows OS specifics.
Android applications consist of bytecode running on the ART virtual machine. Android OS offers two main approaches of application development: using Java programming language SDK, or C/C ++ programming language NDK.
SDK (Software Development Kit) is a set of classes and libraries for Java programming language allowing to utilize the capabilities of Android OS.
NDK (Android Native Development Kit) is a package of tools and libraries allowing to develop applications in C/C++. NDK is recommended for developing code sections, where performance is crucial.
Assuming that the developed application does not require high performance, the best solution would be to use Java SDK.
To make the development process more comfortable, certain development environments (IDE) are used. Two major IDEs are currently used for Android development — Eclipse with ADT plugin (Android Development Tools) and Android Studio. For a long time, Eclipse ADT plugin development was held by Google, but after the release of Android Studio IDE in August 2015, the company stopped supporting this plugin, which means that Eclipse can no longer provide full support for Android OS application development and debugging. Therefore, using Android Studio IDE is a natural solution in this situation.
Design patterns are an essential part of any software. They help keep the code clean, make it scalable and testable. Many design patterns are used in Android development, and we will consider such patterns as: MVVM, MVC, MVP.
MVVM (Model-View-View Model). This approach allows mapping user interface elements to properties and events of the View model. The idea behind MVVM is that each layer of this pattern is not aware about the existence of another layers (https://en.wikipedia.org). MVVM is designed for team-based application development, i.e. while one team member is working on the layout and screen styles, the other can simultaneously code the data reception and processing logic (Figure 3).

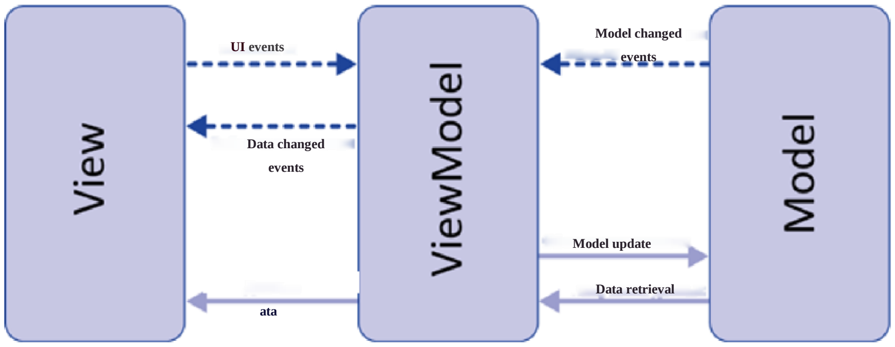


Figure 3: MVVM design pattern diagram

MVC design pattern (Model-View-Controller) has been created to separate business logic from user interface (https://ru.wikipedia.org). The idea behind this design pattern is that the user interacts with the controller, the controller requests data from the model and sends it into View, which is displayed to the user (Figure 4).

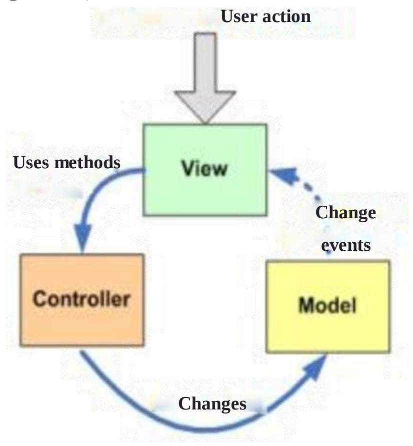


Figure 4: MVC design pattern diagram

One of the most efficient design patterns is MVP (Model-View-Presenter) (https://ru.wikipedia.org), since it allows separating user interface (View) from the data itself (Model), while providing interaction between those via a "middleman" (Presenter), which simplifies Unit test creation and application scaling (Figure 5).

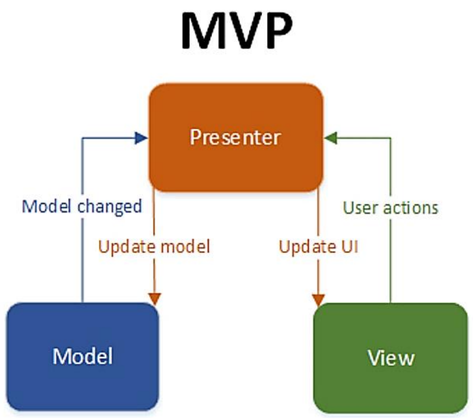


Figure 5: MVP design pattern diagram

This development approach is relatively new, but is rapidly gaining popularity due to its simplicity and efficiency in solving many problems. This design pattern will also be added to the new version of the Java language. The essence of this approach is to use asynchronous data streams. This approach ensures that applications are more resilient to network errors, while simplifying the processing of various events. Therefore, this programming approach has been used in order to increase the application stability.

Before writing this paper, we set a general objective to develop an application that will quickly and efficiently provide users with real-time information about their location and create a route to their destination of travelling. The developed mobile application requires Internet connection via Wi-Fi or GPS. Let us considering the following particular tasks that are solved in the software development process:

– application architectural design;
– receiving data from sensors and Google services;
– map loading;
– filtering and displaying calculation results;
– route creation;
– camera utilization and event display.

After the application is launched, a map view is displayed including the user's current location. Data is received via the Internet connection. In order to include the map in our application, we need to register on the Google Maps website and get the relevant API key.

The application's main screen with a map includes a button (map icon) for route creation. After being touched, 5 markers connected by a line are added to the screen. The markers can be repositioned. The screen also includes buttons to delete the route and switch to the camera view. Further, the user can switch to the camera view by clicking on the button with a camera icon. Switching to another screen does not affect the data about the created route. Data is retrieved from smartphone sensors and via the Internet connection, after which the data is processed and displayed in an information window, as well as in a form of images or pointers. To get the pointer indicate the direction, the angle between the user's coordinates and the map marker shall be calculated. For this, azimuth shall be found, as well as the distance between the user and the map marker. The user sees an arrow on the screen indicating the direction to the first marker on his route. When the user is in the target range, taking into account permissible deviations, the augmented reality object is displayed on the screen, after which the application creates the route to the next point, etc. After the entire route is completed, application switches to the map view, allowing to create a new route.

Software solution architecture

Android application development does not require the developer to use specific hardware and/or software platform, which simplifies the software development process. We have used Google's Android Studio Integrated Development Environment (IDE), which includes all the necessary tools for application development, debugging, and testing. Java 6 is the minimum requirement for working with this IDE and compiling applications, although Java 7 or later is recommended. Android Studio has rich functionality that helps to accelerate application development (https://developer.android.com). This IDE includes a number of necessary tools for application development (Mikheev et al., 2016). To be able to use our application, the user should have a smartphone with a set of necessary sensors, and clean up the smartphone memory cache occasionally to avoid errors inside the application. A review of relevant programming languages is given below.

Justifying programming language choice

Swift is a fast and efficient programming language with real-time response that can be easily inserted into a fully-functional Objective-C code. It eliminates a large layer of common software errors by leveraging modern software design patterns:

- Variables are always initialized before they are used.
- Array indices are always checked for "out-of-bounds" errors.
- Integers are checked for overflow.
- Optional guarantee that nil values will be explicitly processed.
- Automatic memory management.
- Error handling enables controlled recovery from unexpected errors.

C# is an object-oriented language with strong class typing, allowing developers to create safe and reliable applications of all kinds, based on .NET Framework. In addition to mobile application development, C# can also be used to create Windows client applications, XML web services, distributed components, client-server applications, and database applications.
Java is an object-oriented programming language. Initially, the new programming language was called Oak (by James Gosling) and was developed for consumer electronics, but was later renamed to Java and began to be used for writing applets, applications, and server software (Martyshkin et al., 2019).
Java include 3 main technology families:
J2EE — Java Enterprise Edition, for creating industrial-grade software. Figure 6 shows J2EE structure.
J2SE — Java Standard Edition, for creating desktop applications, primarily.
J2ME — Java Micro Edition, for embedding into devices with limited computing power, including mobile phones, PDAs, embedded systems.

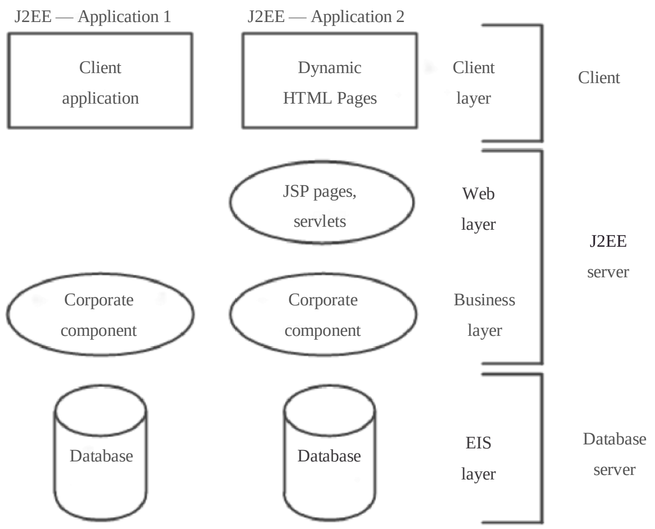


Figure 6: J2EE structure

Java with a set of relevant API libraries is considered the main programming language for Android OS. Android Developers website is regarded the best resource for Android system development guides (Goloshchapov, 2011). Regarding the development IDE — Google has developed a number of additional features for the IntelliJ IDEA IDE and eventually released an IDE called Android Studio. In addition, Java Development Kit (JDK) version 5 or later is required. Out of the abovementioned options, Java programming language is the most suitable for the execution of objectives, as it includes an extensive library of classes, frameworks, etc.

Choice of software development environment
Since we have chosen Java as the programming language, application development will be carried out in Android Studio IDE, being the most relevant for Android development (Martyshkin & Martens-Atyushev, 2019). Each project in Android Studio contains one or more modules with source code files and resource files. The following module types are possible:
– Android application modules.
– Library modules.

– Google App Engine modules.

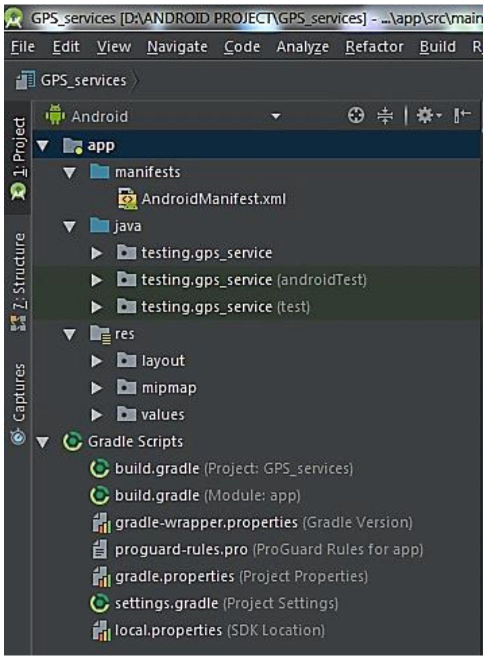
Android Studio displays project files in the Android project view by default (Figure 7). All assembly files are visible at the top level under Gradle Scripts, and each application module contains the following folders:

–manifests — contains AndroidManifest.xml file;

–java — contains Java source code files, including JUnit test code;

–res — contains all non-code assets, such as XML layouts, user interface strings and bitmaps.

–


GPS_services [D:\ANDROID PROJECT\GPS_services] - ...\app\src\main

File Edit View Navigate Code Analyze Refactor Build R

GPS_services

Android

```
app
  manifests
    AndroidManifest.xml
  java
    testing.gps_service
    testing.gps_service (androidTest)
    testing.gps_service (test)
  res
    layout
    mipmap
    values
Gradle Scripts
  build.gradle (Project: GPS_services)
  build.gradle (Module: app)
  gradle-wrapper.properties (Gradle Version)
  proguard-rules.pro (ProGuard Rules for app)
  gradle.properties (Project Properties)
  settings.gradle (Project Settings)
  local.properties (SDK Location)
```

Figure 7: Android Studio project files view

The data structure development is one of the most important tasks at the initial stage of development. Often, the data, in particular, determines the possible functionality of the developed application and its operation principles. MVP design pattern used in this paper allows to separate the data from their presentation, which makes it possible to store data in any convenient form and change it without impairing the main application logic. Input comes from mobile device sensors and Google services. SensorManager class is used to work with hardware sensors. Regardless of how many sensors will be used in the application, all sensors work independently of each other and data from them are easy to acquire. However, working with sensors significantly increase energy consumption, and that fact shall be taken into account.

Typically, real devices are used to test similar applications that utilize hardware sensors, and it is best to ensure that the smartphone have these sensors. This can be done by reviewing the smartphone specifications or by coding a small application to display readings from all sensors. After the data is received, it is presented in the form of a data stream, which contains class structures. Data composition and structure related to route creation with augmented reality are given in Tables 1–5.

Table 1: CameraViewActivity class data composition and structure

Description	Name	Type
Real azimuth	mAzimuthReal	double
Theoretical azimuth	mAzimuthTeoretical	double

Distance permissible deviation	DISTANCE_ACCURACY	double
Azimuth permissible deviation	AZIMUTH_ ACCURACY	double
Device latitude	mMyLatitude	double
Device longitude	mMyLongitude	double
Arrow rotation angle	RotateDegree	float
Number of points travelled	ArValue	int
Device and AR object latitude difference	dX	double
Device and AR object longitude difference	dY	double
Distance to AR object	distance	double
Azimuth accuracy for object display	minMax	ArrayList<double>
Minimum azimuth angle	minAngle	double
Marker A latitude	TargetLatMarA	double
Marker A longitude	TargetLngMarA	double
Minimum azimuth angle	mixAngle	double
Maximum azimuth angle	maxAngle	double
Marker B latitude	TargetLatMarB	double
Marker B longitude	TargetLngMarB	double
Marker C latitude	TargetLatMarC	double
Marker C longitude	TargetLngMarC	double
Marker D latitude	TargetLatMarD	double
Marker D longitude	TargetLngMarD	double
Marker E latitude	TargetLatMarE	double
Marker E longitude	TargetLngMarE	double
Main information on the screen	textData	String

Table 2: MapActivity class data composition and structure

Description	Name	Type
Marker A	mMarkerA	Marker
Marker B	mMarkerB	Marker
Marker C	mMarkerC	Marker
Marker D	mMarkerD	Marker
Marker E	mMarkerE	Marker
Device latitude	mMyLatitude	double
Device longitude	mMyLongitude	double
Marker A latitude	marALat	double
Marker A longitude	marALng	double
Marker B latitude	marBLat	double
Marker B longitude	marBLng	double
Marker C longitude	marCLng	double
Marker D latitude	marDLat	double
Marker D longitude	marDLng	double
Marker E latitude	marELat	double
Marker E longitude	marELng	double
Button operation variable	clearValue	int
Marker C latitude	marCLat	double

Table 3: MyCurrentAzimuth class data composition and structure

Description	Name	Type
Rotation angle degree	azimuthFrom	int
Rotation angle degree	azimuthTo	int

Table 4: MyCurrentLocation class data composition and structure

Description	Name	Type
Object latitude	mLatitude	double
Object longitude	mLongitude	double

Table 5: ArObject class data composition and structure

Description	Name	Type
Object latitude	mLatitude	double
Object longitude	mLongitude	double

All data received from the services are stored in the local storage, which allows to retrieve data from the local storage and continue working with it when Internet connection is lost or in case of a network error. The data from sensors do not require long-term storage, as it is constantly changing.

Algorithm development

After launching the application, the user needs to allow detection of his phone location, after which his location is detected and the map view is centred at it. If the location cannot be detected, then data on the user's last location will be retrieved from storage, in other case a message will be displayed on the screen with a hint. A route can be created after that, which can be modified or deleted as well. Further, user can touch the button with a camera icon to switch to the camera view. If the user has committed some wrong action, then a message will be displayed with a hint at the bottom of the screen. More details about such messages are provided further in the text.
The user can freely switch between the camera and map views, however, if the user has not travelled the entire route and returned to the map view, he will have to create the route again, since the route is not saved. Screen navigation diagram is shown in Figure 8.

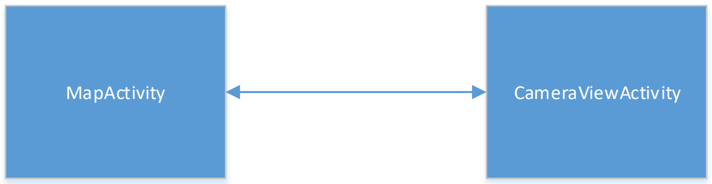


Figure 8: Screen navigation diagram

When the user switches from the map view to the camera view, the route data is passed using Intent method, after which the received data is recorded in predefined variables and used within the code. Camera and map classes use data obtained from sensors, which is done by running classes created to receive readings from sensors. To initialize the classes, start () method is called and an interface is implemented to pass the data.
Immediately after switching to the map view, data retrieved from the Internet and device sensors is processed, after which the screen is loaded with the camera view displayed and the direction pointer to the first route marker. The pointer is created by mapping the orientation sensor to the pointer, after which the azimuth is calculated by the formula (1). The azimuth value is required in order to find the angle between the user and marker on the view.

$$\tan \varphi_{AB} = \left| \frac{y_{AB}}{x_{AB}} \right|, \qquad (1)$$

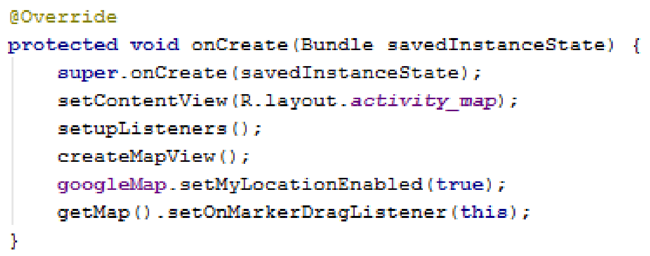
To be able to properly display the augmented reality object, both the azimuth and distance values shall be known. The distance is calculated by the formula (2).

$$d_{AB} = \sqrt{\Delta x^2 + \Delta y^2}, \tag{2}$$

To make the application more user friendly, the code includes constants for permissible azimuth and distance deviations, which are used to make the object easier to display on the camera view. When the user is in the target range, taking into account permissible deviations, the display condition is satisfied and the augmented reality object is displayed on the screen, after which the application receives coordinates for the next marker and performs the same operations with it. After the entire route is completed, application switches to the map view, allowing to create a new route.

Application development

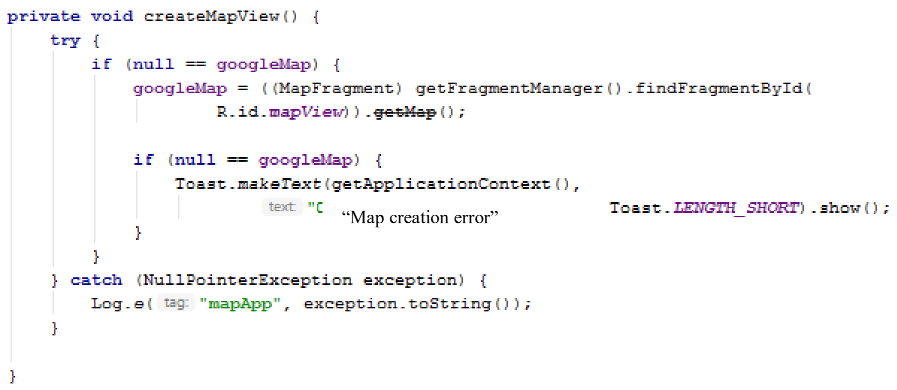
The developed AR Navigator mobile application is intended to show directions to the destination object using a map and camera with augmented reality. The application shall be written in the Java programming language and have the functionality specified in the beginning of this paper. The application starts with MapActivity class. When this class is initialized, onCreate() method is called, which initializes the map, connects to Google services, requests access to the current location and triggers the map marker handling method (Figure 9).



```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_map);
    setupListeners();
    createMapView();
    googleMap.setMyLocationEnabled(true);
    getMap().setOnMarkerDragListener(this);
}
```

Figure 9: OnCreate() method of MapActivity class

To add a map to the fragment, createMapView() method is used, which connects to Google services and displays the map in case of a successful connection to the Internet (Figure 10). An error message is displayed if the map could not be created. After the map is added, route creation can be initiated. To do this, the application needs to know the user’s coordinates and add markers connected by a line, get the latitude and longitude of the device and assign their values.



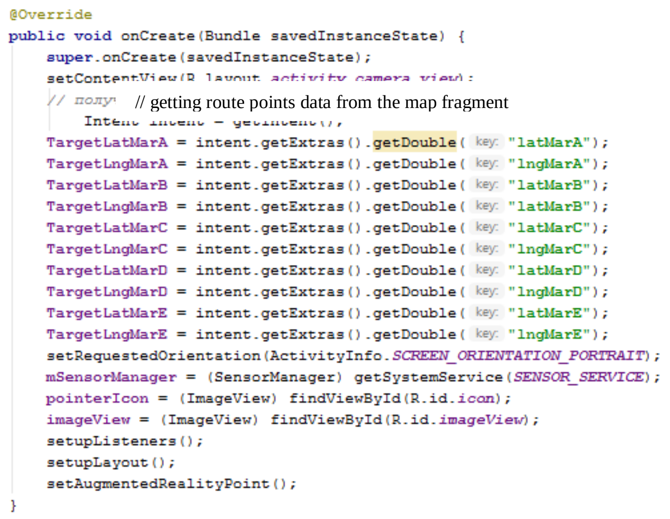
```
private void createMapView() {
    try {
        if (null == googleMap) {
            googleMap = ((MapFragment) getFragmentManager().findFragmentById(
                    R.id.mapView)).~~getMap~~();

            if (null == googleMap) {
                Toast.makeText(getApplicationContext(),
                        text: "C  “Map creation error”          Toast.LENGTH_SHORT).show();
            }
        }
    } catch (NullPointerException exception) {
        Log.e( tag: "mapApp", exception.toString());
    }

}
```

Figure 10: createMapView () method

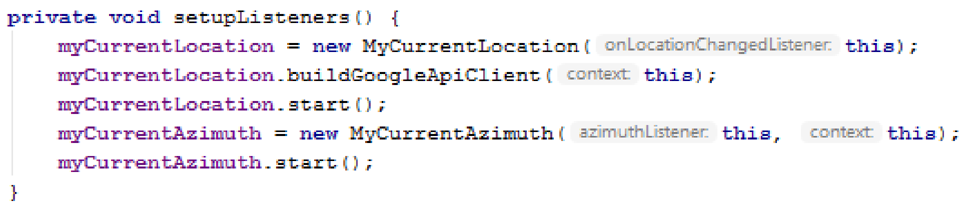
CameraViewActivity class implements the main idea of the application. When switching to this Activity, OnCreate() method is triggered (Figure 11), after which the following chain of events occurs: route data is received from MapActivity class, MyCurrentAzimuth and MyCurrentLocation class constructors are called and their start() methods executed, screen elements and camera view are initialized, an augmented reality object instance is created.



```
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_camera_view);
    // полу  // getting route points data from the map fragment
        Intent intent = getIntent();
    TargetLatMarA = intent.getExtras().getDouble( key: "latMarA");
    TargetLngMarA = intent.getExtras().getDouble( key: "lngMarA");
    TargetLatMarB = intent.getExtras().getDouble( key: "latMarB");
    TargetLngMarB = intent.getExtras().getDouble( key: "latMarB");
    TargetLatMarC = intent.getExtras().getDouble( key: "latMarC");
    TargetLngMarC = intent.getExtras().getDouble( key: "lngMarC");
    TargetLatMarD = intent.getExtras().getDouble( key: "latMarD");
    TargetLngMarD = intent.getExtras().getDouble( key: "lngMarD");
    TargetLatMarE = intent.getExtras().getDouble( key: "latMarE");
    TargetLngMarE = intent.getExtras().getDouble( key: "lngMarE");
    setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_PORTRAIT);
    mSensorManager = (SensorManager) getSystemService(SENSOR_SERVICE);
    pointerIcon = (ImageView) findViewById(R.id.icon);
    imageView = (ImageView) findViewById(R.id.imageView);
    setupListeners();
    setupLayout();
    setAugmentedRealityPoint();
}
```

Figure 11: OnCreate method of CameraViewActivity class

setupListener() method creates all the elements on the screen and initiates the camera (Figure 12).



```
private void setupListeners() {
    myCurrentLocation = new MyCurrentLocation( onLocationChangedListener: this);
    myCurrentLocation.buildGoogleApiClient( context: this);
    myCurrentLocation.start();
    myCurrentAzimuth = new MyCurrentAzimuth( azimuthListener: this,  context: this);
    myCurrentAzimuth.start();
}
```

Figure 12: setupListener ()method of CameraViewActivity class

Objects package classes describe the data structures received by the application from the services and sensors they interact with:

– ArObject — describes augmented reality object data;
– MyCurrentAzimuth — determines the azimuth;
– MyCurrentLocation — detects the device location;

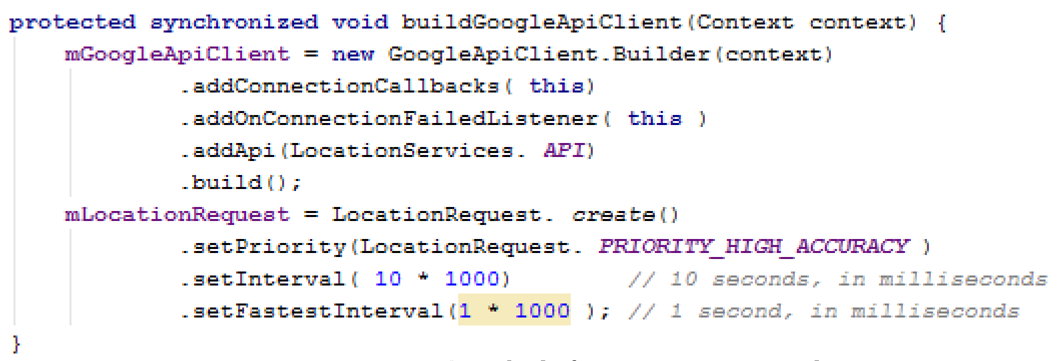
onAzimuthChangedListener interface describes a method for azimuth calculation.

  – request method;
  – passed attributes;

onLocationChangedListener interface describes a method for device coordinates handling and includes:

  – request method;
  – passed attributes;

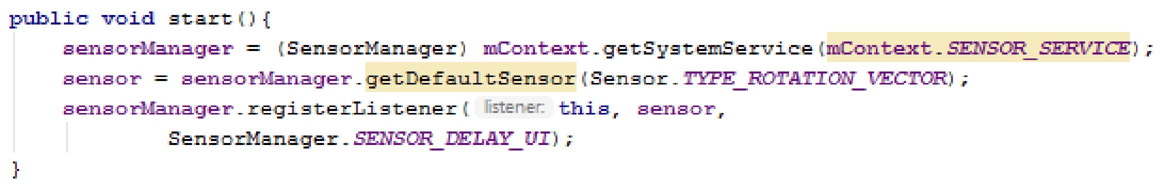
MyCurrentLocation class includes buildGoogleApiClient method, which creates ApiClient that uses .addApi method to call the location discovery. A request is created and the interval for request sending is set (Figure 13).



```
protected synchronized void buildGoogleApiClient(Context context) {
    mGoogleApiClient = new GoogleApiClient.Builder(context)
            .addConnectionCallbacks( this)
            .addOnConnectionFailedListener( this )
            .addApi(LocationServices. API)
            .build();
    mLocationRequest = LocationRequest. create()
            .setPriority(LocationRequest. PRIORITY_HIGH_ACCURACY )
            .setInterval( 10 * 1000)        // 10 seconds, in milliseconds
            .setFastestInterval(1 * 1000 ); // 1 second, in milliseconds
}
```

Figure 13: start() method of MyCurrentLocation class

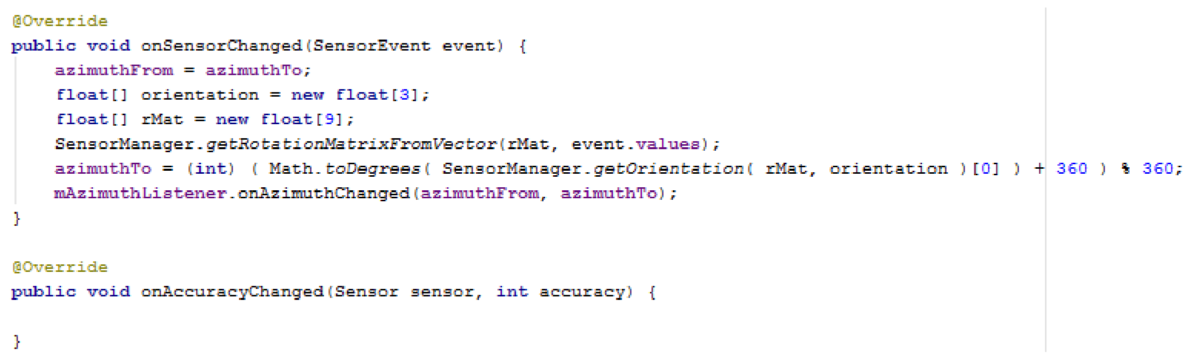
start() method of MyCurrentAzimuth class connects to a sensor and registers a sensor listener with a specified refresh rate. Data is collected from the accelerometer, gyroscope and magnetic field sensor. After being called, the method is triggered every time the phone moves in space (Figure 14).



```
public void start(){
    sensorManager = (SensorManager) mContext.getSystemService(mContext.SENSOR_SERVICE);
    sensor = sensorManager.getDefaultSensor(Sensor.TYPE_ROTATION_VECTOR);
    sensorManager.registerListener( listener: this, sensor,
            SensorManager.SENSOR_DELAY_UI);
}
```

Figure 14: start() method of MyCurrentAzimuth class

To process and output data from sensors, SensorEventListener interface with the two methods onSensorChanged() and onAccuracyChanged() are implemented. Data is received via an asynchronous stream (Figure 15).
onSensorChanged() method is called when a new sensor event occurs, and a device rotation matrix is retrieved.



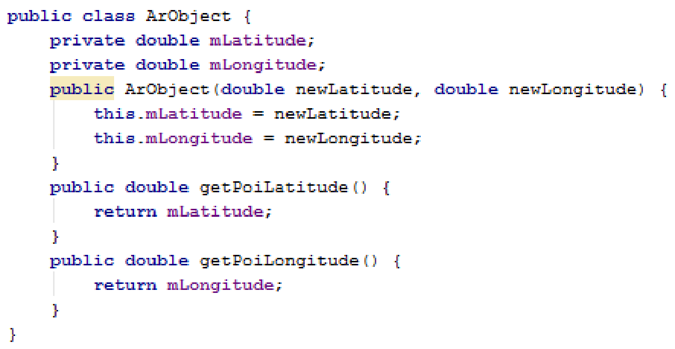
```
@Override
public void onSensorChanged(SensorEvent event) {
    azimuthFrom = azimuthTo;
    float[] orientation = new float[3];
    float[] rMat = new float[9];
    SensorManager.getRotationMatrixFromVector(rMat, event.values);
    azimuthTo = (int) ( Math.toDegrees( SensorManager.getOrientation( rMat, orientation )[0] ) + 360 ) % 360;
    mAzimuthListener.onAzimuthChanged(azimuthFrom, azimuthTo);
}

@Override
public void onAccuracyChanged(Sensor sensor, int accuracy) {

}
```

Figure 15: SensorEventListener interface implementation

ArObject class contains a set of variables for storing marker coordinates, as well as methods for retrieving getPoiLatidude() and getPoiLongitude() (Figure 16).

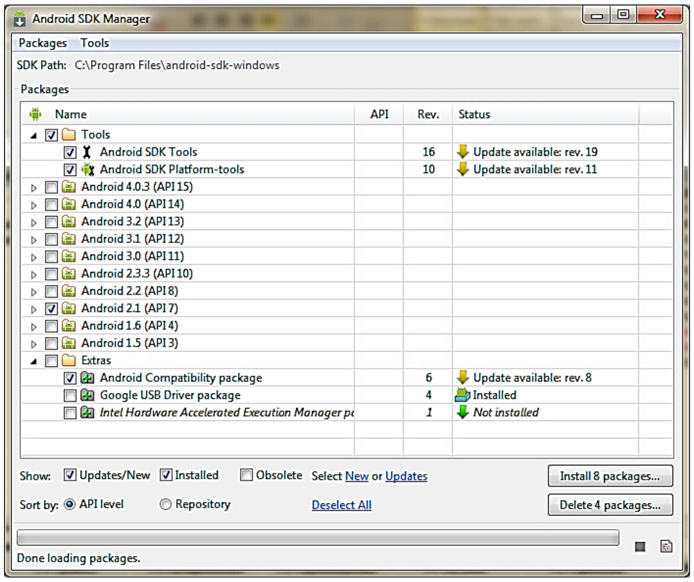


```
public class ArObject {
    private double mLatitude;
    private double mLongitude;
    public ArObject(double newLatitude, double newLongitude) {
        this.mLatitude = newLatitude;
        this.mLongitude = newLongitude;
    }
    public double getPoiLatitude() {
        return mLatitude;
    }
    public double getPoiLongitude() {
        return mLongitude;
    }
}
```

Figure 16: ArObject class

Data processing and displaying on the screen occurs in Activity classes, after which the collected data is passed to the required class by calling methods from the data collection classes.
The following is required to ensure correct functioning of the application:
– smartphone with Internet connection and GPS;
– relevant mobile device sensors.

To use the application, a real device running Android 4.3 or later, or a virtual machine with similar specifications is needed.
If there is no real device, the application can be installed and tested on an Android OS emulator. For this, install the basic SDK toolkit from the website (Goloshchapov, 2011). Unpack the downloaded archive into C:\Program Files\android-sdk folder. Run SDK Manager.exe file from this folder. The window will be displayed as shown in Figure 17.



Android SDK Manager

Packages Tools

SDK Path: C:\Program Files\android-sdk-windows

Packages

Name	API	Rev.	Status
✓ Tools			
✓ Android SDK Tools		16	Update available: rev. 19
✓ Android SDK Platform-tools		10	Update available: rev. 11
☐ Android 4.0.3 (API 15)			
☐ Android 4.0 (API 14)			
☐ Android 3.2 (API 13)			
☐ Android 3.1 (API 12)			
☐ Android 3.0 (API 11)			
☐ Android 2.3.3 (API 10)			
☐ Android 2.2 (API 8)			
✓ Android 2.1 (API 7)			
☐ Android 1.6 (API 4)			
☐ Android 1.5 (API 3)			
☐ Extras			
✓ Android Compatibility package		6	Update available: rev. 8
☐ Google USB Driver package		4	Installed
☐ *Intel Hardware Accelerated Execution Manager pa*		*1*	*Not installed*

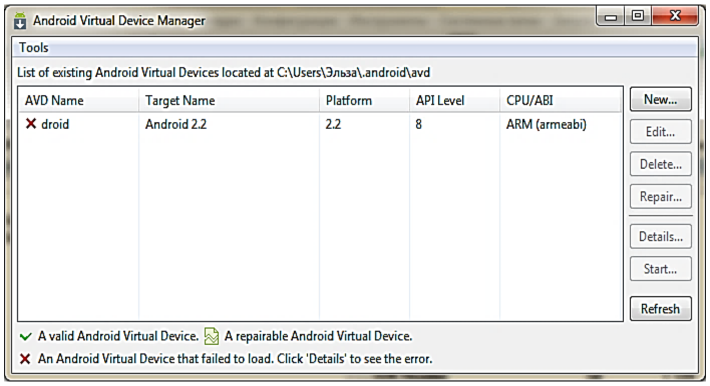
Show: [✓ Updates/New] [✓ Installed] ☐ Obsolete Select New or Updates   Install 8 packages...

Sort by: [✓ API level] ○ Repository   Deselect All   Delete 4 packages...

Done loading packages.

Figure 17: Android SDK Manager

In this window, tick Tools, Android 4.0.3 (API15) folder. Click Install button. After the toolkit is installed, create an Android virtual device. To do this, launch Manager.exe from C:\Program Files\android-sdk folder. The window will be displayed as shown in Figure 18.


Android Virtual Device Manager

Tools

List of existing Android Virtual Devices located at C:\Users\Эльза\.android\avd

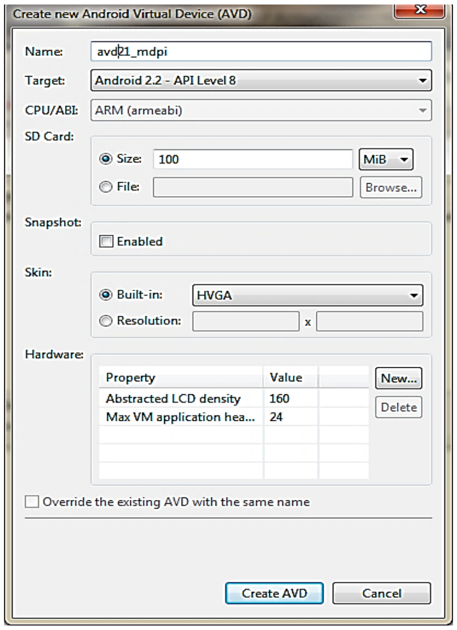
AVD Name	Target Name	Platform	API Level	CPU/ABI
✗ droid	Android 2.2	2.2	8	ARM (armeabi)

✓ A valid Android Virtual Device. A repairable Android Virtual Device.

✗ An Android Virtual Device that failed to load. Click 'Details' to see the error.

Figure 18: Android Virtual Device Manager

Click New button to create a new virtual device. The dialog window will be displayed as shown in Figure 19.


Create new Android Virtual Device (AVD)

Name: avd21_mdpi

Target: Android 2.2 - API Level 8

CPU/ABI: ARM (armeabi)

SD Card:
- ● Size: 100 MiB
- ○ File:

Snapshot:
- ☐ Enabled

Skin:
- ● Built-in: HVGA
- ○ Resolution: x

Hardware:

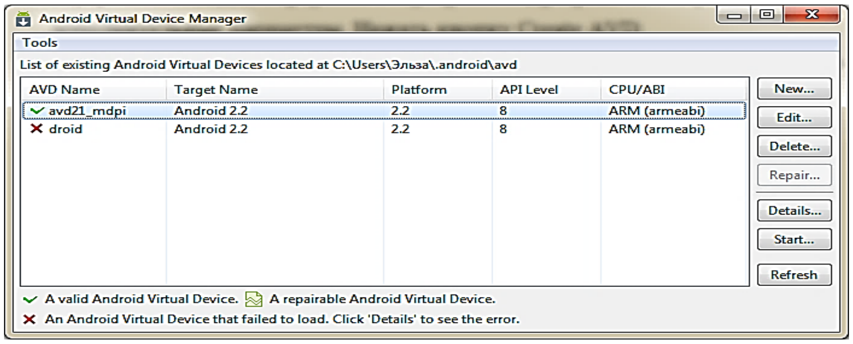
Property	Value
Abstracted LCD density	160
Max VM application hea...	24

☐ Override the existing AVD with the same name

Figure 19: Creating a new virtual device

A dialog is displayed, prompting to specify the name of the virtual device, Android OS version (at least 4.0.3), virtual device internal memory size, screen resolution and other additional parameters. Click Create AVD button.

To start the virtual device, select it from the list of available devices and click Start button (Figure 20).


Android Virtual Device Manager

Tools

List of existing Android Virtual Devices located at C:\Users\Эльза\.android\avd

	AVD Name	Target Name	Platform	API Level	CPU/ABI
✓	avd21_mdpi	Android 2.2	2.2	8	ARM (armeabi)
✗	droid	Android 2.2	2.2	8	ARM (armeabi)

✓ A valid Android Virtual Device. A repairable Android Virtual Device.

✗ An Android Virtual Device that failed to load. Click 'Details' to see the error.

Figure 20: Starting a virtual device

After the device boots up (Figure 21), you can proceed to install the application. The developed application file is called AugmentedRealityNavigator.apk.

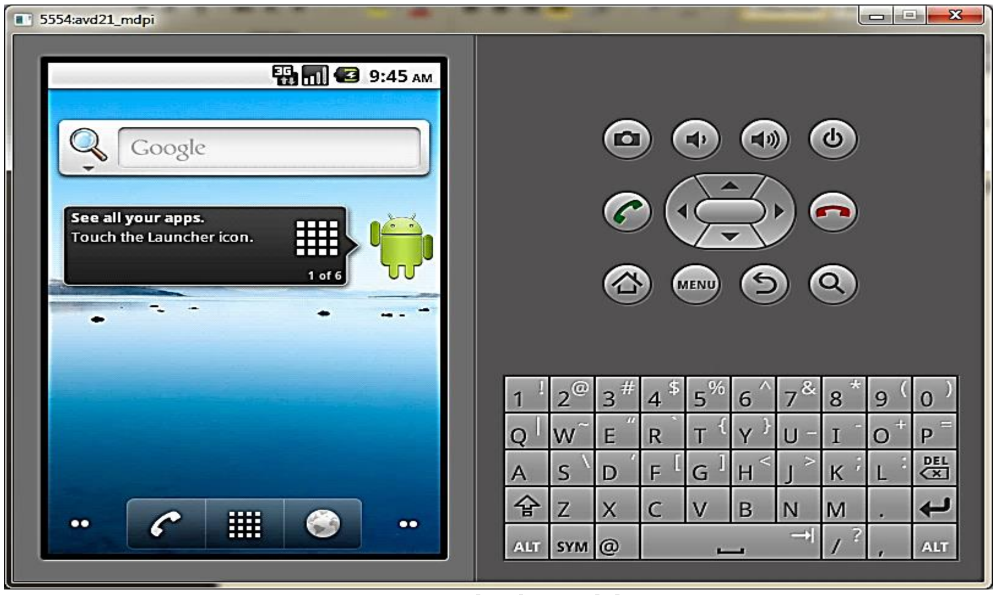


Figure 21: Android virtual device

To do this, run the command line, change to C:\Program Files\android-sdk\platform-tools folder, and execute the following command:

adb install –r <file_name>,

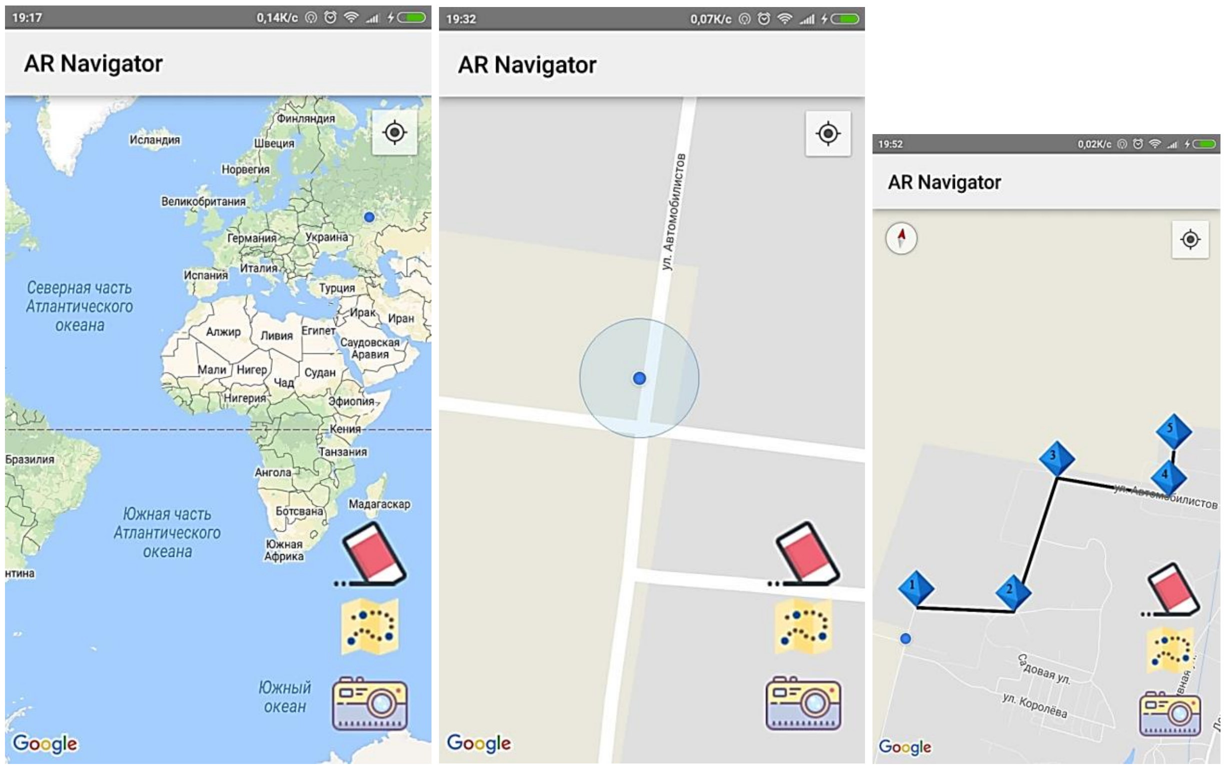
where <file_name> shall be replaced with the full path to AugmentedRealityNavigator.apk file. The –r parameter indicates that if the application was installed on the device before, it will be deleted first and then installed again.

The application can be launched by selecting the application icon from the list of installed applications.

User instructions

AR Navigator mobile application is a map directions information system utilizing augmented reality technology. Before starting, the device shall be connected to the Internet and with GPS enabled, to enable the map view and obtain device location data. In case there is a problem with the map and it is not drawn, a relevant message will be displayed (Message No. 1, Table 6). After launching, the main application screen is displayed (Figure 22, a). This screen contains a built-in

Google map view, as well as the application control buttons. By touching My Location button in the upper right corner, the user’s location data is downloaded, and the map view is centred at the current location. If the data could not be loaded, the user's last known location will be displayed. The user's location is displayed on the map as a small blue dot (Figure 22, b).
The lower part of the screen has 3 essential buttons, namely: create a route, delete a route, switch to augmented reality mode.
By touching Create Route button with a map icon, 5 markers are added to the map near the user's location, connected by a line. These markers have numbers and can be moved (Figure 22, c). If the route has already been created, a relevant message will be displayed on the screen (Message No. 5, Table 6).
An augmented reality pointer is mapped to each marker. If the user's location has not been determined, a message will be displayed on the screen (Message No. 2, Table 6).
By touching Delete Route button with an eraser icon, a check is performed to find any connected markers on the map, which are deleted along with the line connecting them if found. If no route is found on the map, a relevant message is displayed (Message No. 3, Table 6).

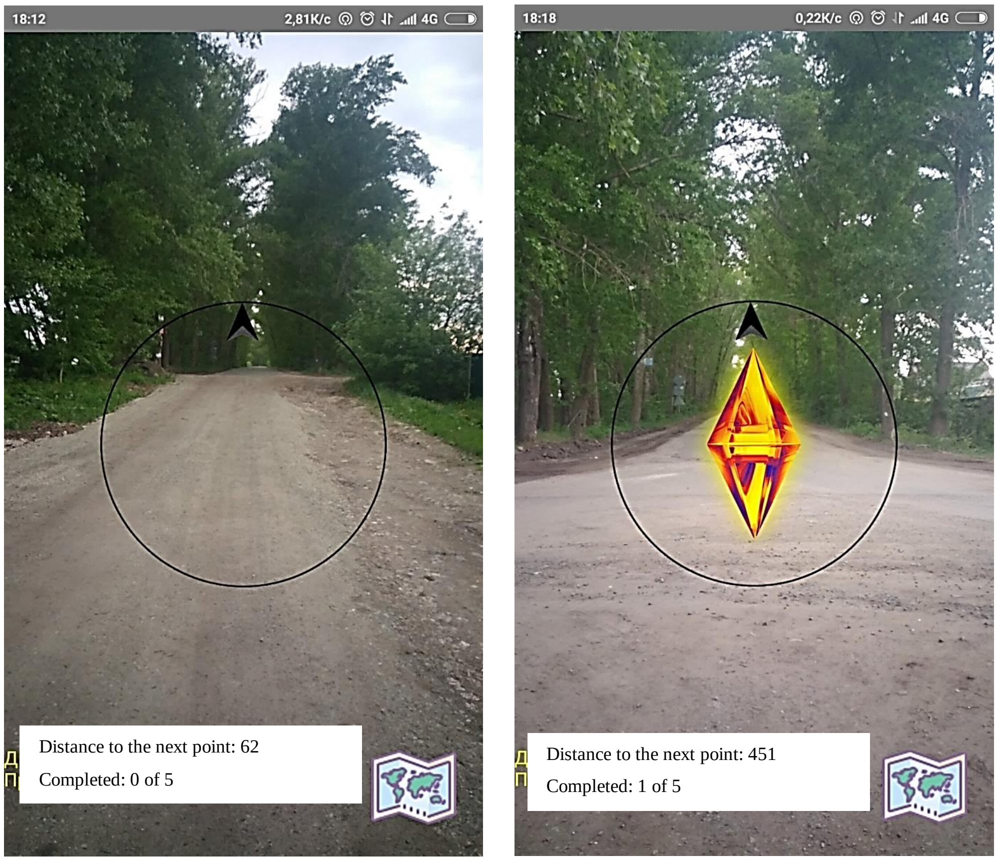


a) b) c)

Figure 22: AR Navigator application's main screen (a); map with device location displayed (b); map with a created route (c)

By touching button Switch to Augmented Reality Mode with a camera icon, the application switches to a new window with a camera view (Figure 23, a). Before touching this button, a route shall be created first. If no route exists, a relevant message will be displayed on the screen (Message No. 4, Table 6).
After switching to Augmented Reality Mode, a circle with an arrow will appear on the screen, showing the direction to the first of five points on the route. The bottom of the screen displays

how many route points the user have travelled, the distance to the next point and a button to switch to the map view. When the user arrives at the destination, the marker image appears on the screen in an augmented reality mode (Figure 23, b).



Distance to the next point: 62
Completed: 0 of 5

Distance to the next point: 451
Completed: 1 of 5

a) b)
Figure 23: Augmented reality mode (a); AR object displayed (b)

After the augmented reality object is displayed, the bottom screen information is updated and the route to the next point is created, after which the AR object remains on the screen for a while, and then disappears. When the user arrives at the last point, the augmented reality loop will end and the application will automatically switch to the map view, allowing the user to create a new route.

Table 6: Error Messages

Message No.	Message text	Cause	User action
1	Map generation error	Failed to connect to Google services	Disconnect and connect again
2	Wait for your location to be detected	Failed to load device location data	Disconnect and connect again
3	Cannot delete route	No route has been added	Nothing
4	Create route!	Create a route before switching to AR mode.	Nothing
5	Route already created!	Route cannot be created, as it has already been created	Nothing

Below are minimum system requirements for the application to work correctly:
– Smartphone running Android 4.3 or later;
– CPU clock frequency — at least 400 MHz;
– RAM size — at least 512 MB;
– free internal space — at least 100 MB;
– Internet connection;
– active GPS;
– camera;
– compass, accelerometer, gyroscope, magnetic field sensor.

The main input data for the application is data received via the Internet connection and GPS, as well as readings collected from sensors.

To use the proposed application, the user should have a basic knowledge of operating cell phones with a touch screen. All interaction with the device is done via a touch screen.

The approaches used in the application development remove a number of operations required for application maintenance. But in order to ensure stable operation, it is recommended to clean RAM and internal memory cache of the device, check its operation, as well as connection to the Internet and GPS status.

The application was developed and tested using Xiaomi Redmi 4 with the following specifications:
– Android OS 6.0.1;
– CPU clock frequency — 2260 MHz;
– RAM size — 3 GB;
– free internal space — 32 GB.

CONCLUSIONS

Based on the results of the research, this paper presents a developed system, implementing the functions specified in the research objective. The application allows using mobile devices for quick and efficient route creation, and includes augmented reality elements.

The pros of the system include utilization of promising mobile technologies that are gaining widespread popularity, code succinctness, simplicity of use on a regular mobile phone, and an intuitive user interface.

The fact that the system is running Android OS installed on a mobile device can be seen as an advantage as well. Not every mobile phone is based on this OS, but, namely, this platform is currently the most popular one. The advantage of Android OS is that phones of different price categories are produced to run under this OS. The distribution of the proposed application and its implementation are also quite easy, since Android OS developers have provided a special Internet resource for distribution of both free and paid applications.

This application can be extended and enhanced by developing versions for other mobile operating systems (for example, iOS and Windows Phone).

3. ACKNOWLEDGMENT

The article is published with the support of the scholarship from the President of Russian Federation to young scientists and graduate students in 2018-2020 (SP-68.2018.5).

REFERENCES

Android [Electronic source] – Official Android WebSite. – Available at: https://www.android.com/intl/ru_ru/

Android Developer [Electronic source] – AndroidDeveloper Website article. – Available at: https://developer.android.com/studio/features.html

Android Developer [Electronic source] – Official Android Developer WebSite. – Available at: http://developer.android.com/

Azuma, R.T. (1997). A Survey of Augmented Reality. Presence: Teleoperators and Virtual Environments, 6(4), 355–385.

Boychenko, I.V., & Lezhankin, A.V. (2010). Augmented reality: current state, problems and solutions. Proceedings of Tomsk State University of Control Systems and Radioelectronics, 1-2(21), 161-165.

Caudell, T.P., & Mizell, D.W. (1992). Augmented reality: an application of heads-up display technology to manual manufacturing processes. Proceedings of the Twenty-Fifth Hawaii International Conference on System Sciences, 7–10 Jan. 1992. URL: https:// ieeexplore.ieee.org/document/183317/.

Chirkin, A.N., & Pimonov, A.G. (2019). Presentation of the company’s capabilities using the augmented reality application. Innovations in information technology, engineering and motor transport Proceedings of the III International Scientific and Practical Conference. Editorial board: D.M. Dubinkin [et al.], 101-102.

Deitel, P., Deitel, H., Deitel, E., & Morgan, M. (2012). Android for programmers. An app-driven approach. Peter, - 560 p.

Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (2016). Design Patterns: Elements of Reusable Object-Oriented Software Peter, – 432 p.

Goloshchapov, A.L. (2011). Google Android: Mobile Programming. BHV-Petersburg,. - 804 p.

Google Design [Electronic source] – Official Google Design WebSite. – Available at: https://design.google.com

Google Maps API [Electronic source] – Google Maps – Available at: https://developers.google.com/maps/

Google Maps API [Electronic source] – Google Maps Android API Utility Library – Available at: https://developers.google.com/maps/documentation/android-api/utility/?hl=ru

Honcharova, N. (2019). Technology of Augmented Reality in Textbooks of New Generation. Problems of modern textbooks, 22, 46-56.

Ivanova, A. (2018). Virtual and augmented reality technologies: opportunities and holdbacks of application. Strategic decisions and risk management, 3(108), - ISSN 2618-947X

JetBrains [Electronic source] – Official JetBrains WebSite. – Available at: https://plugins.jetbrains.com/

Klimov's, A. Website [Electronic source] – Acquiring data from sensors – Available at: http://developer.alexanderklimov.ru/android/sensors.php

Kuraev, N.I. (2019). Digitalization and augmented reality applications. 21st Century Youth: Education, Science, Innovation: Proceeding of the VIII All-Russian Student Scientific and Practical Conference with international participation. Edited by T.A. Biryukova, 15-16.

Martyshkin, A.I., & Martens-Atyushev, D.S. (2019). Mathematical modelling and evaluation of the characteristics of specialized reconfigurable systems based on a common bus at the stage of synthesis of the system configuration. Journal of Advanced Research in Dynamical and Control Systems, 11(8 Special Issue), 2852-2860

Martyshkin, A.I., Pashchenko, D.V., & Trokoz, D.A. (2019). Queueing Theory to Describe Adaptive Mathematical Models of Computational Systems with Resource Virtualization and Model Verification by Similarly Configured Virtual Server. Proceedings - 2019 International Russian Automation Conference, RusAutoCon 2019, 8867620.

Mayer, R. (2011). Professional Android 2 Application Development. Eksmo, - 671 p.

McC Smith, J. (2013). Elemental design patterns. Penza, - 23 p.

Mikheev, M.Yu., Zhashkova, T.V., Meshcheryakova, E.N., Gudkov, K.V., & Grishko, A.K. (2016). Imitation modelling for the subsystem of identification and structuring data of signal sensors. Proceedings of 2016 IEEE East-West Design and Test Symposium, EWDTS 2016. no. 7807748

Milgram, P., & Kishino, F. (1994). A taxonomy of mixed reality visual displays. IEICE Transactions on Information and Systems, E77-D(12), 1321–1329.

Pavlova, K.T., Faleyeva, Ye.V., & Pavlov, N.G. (2020). Aspects of approaches of user interaction with augmented reality interfaces. Fundamental and Applied Scientific Research: Actual Issues, Achievements and Innovations Collection of articles of the XXXI International Scientific and Practical Conference, 92-94.

Statcounter [Electronic source] – Statcounter article. – Last access date: https://gs.statcounter.com/os-market-share

Tsvetkov, V.Ya. (2017). Augmented Reality. International Journal of Applied and Basic Sciences, 6-2, 211-212.

Wearesocial [Electronic source] – Wearesocial Website article. – Available at: https://wearesocial.com/blog/2020/04/digital-around-the-world-in-april-2020

Wikipedia [Electronic source] – Wikipedia article. – Available at: https://en.wikipedia.org/wiki/Application_programming_interface

Wikipedia [Electronic source] – Wikipedia article. – Available at: https://ru.wikipedia.org/wiki/Material_Design

Wikipedia [Electronic source] – Wikipedia article. – Available at: https://en.wikipedia.org/wiki/Azimuth

Wikipedia [Electronic source] – Wikipedia article. – Available at: https://en.wikipedia.org/wiki/Model–view–viewmodel

Wikipedia [Electronic source] – Wikipedia article. – Available at: https://ru.wikipedia.org/wiki/Model-View-Controller

Wikipedia [Electronic source] – Wikipedia article. – Available at: https://ru.wikipedia.org/wiki/Model-View-Presenter

Yakovlev, B.S., & Pustov, S.I. (2013). Classification and prospects of augmented reality technology. News of the Tula State University. Technical sciences, 3, 484-492.

Yakovlev, B.S., & Pustov, S.I. (2013). History, specific aspects and prospects of augmented reality technology. News of the Tula State University. Technical sciences, 3, 479-484.

Yegorov, A.A. (2019). One of the modern problems of augmented reality in portable devices. 21st Century Youth: Education, Science, Innovation: Proceedings of the VIII All-Russian Student Scientific and Practical Conference with international participation. Edited by T.A. Biryukova, 13-14.