



jamk.fi

Learning programming with interactive Android application

Csaba Nándor Szilágyi

Bachelor's thesis

May 2020

Programme of Information and Communication Technology

Jyväskylän ammattikorkeakoulu

JAMK University of Applied Sciences

Author(s) Csaba Nándor Szilágyi	Type of publication Bachelor's thesis	Date May 2020 Language of publication: English
	Number of pages 39	Permission for web publication: x
Title of publication Learning programming with interactive Android application		
Degree programme Programme of Information and Communication Technology		
Supervisor(s) Manninen, Pasi & Salmikangas, Esa		
Assigned by University of Debrecen		
Abstract <p>The aim of this study was to learn to use the Android Studio and prepare an application which could offer a novelty. The mobiles are nearly as advanced as some computers and a programming environment or a programming assistant app could not be found to the mobile platform which would enable user-friendly coding.</p> <p>In the first steps, existing similar programs were collected, familiarized with and compared. These programs provided some opportunities, for example, the operation, function and objectives that could help the development.</p> <p>In the next step, the designs, plans and implementations were made, e.g. the user interface, objects, interactive objects, lines, side menu, class, function, main menu options and submenu options. Finally, some example programs were created to help with the use and understanding.</p> <p>In a Hungarian national scientific competition, the app was entered into the "learning methodology section" and it took first place.</p> <p>The actual created application tries to help programming on the mobile platform in an interactive way. It could be used by an amateur, a learner or even by someone who has some knowledge about coding. The main aim of this report is to demonstrate a major project initiation and implementation for beginner programmers who want to learn a new technology and create software with it.</p>		
Keywords/tags (subjects) Android Studio, Application, IDE, Java, Mobile, Mobile Development, Programming		
Miscellaneous		

Contents

1	Introduction	4
2	Basics of Android software development	5
2.1	Researching technologies.....	5
2.2	Android Studio development environment	6
2.3	Test environment	7
2.4	Run on real devices	8
2.5	Elements of Android application	11
2.5.1	Activity	11
2.5.2	Fragment.....	12
3	Useful accessories for the developing	12
3.1	Debug running application	12
3.2	Publishing applications.....	13
3.2.1	A way to publish applications.....	13
3.2.2	Developer Certificate.....	13
3.3	Tools used.....	14
4	Similar applications	17
4.1	Researching similar opportunities.....	17
4.2	Introducing Blockly	17
4.3	Introducing Scratch	18
4.4	Introducing Code.org.....	19
4.5	Comparison	20
5	Presentation of application.....	21
5.1	User interface	21

	2
5.2	Menu Bar22
5.3	Integer and Boolean variables.....23
5.4	Operators.....25
5.5	Presentation of For, While and Do-While cycles.....26
5.6	Conditions.....27
5.7	Reference and Value Transfer28
5.8	Example Programs.....29
5.8.1	Variables and operators29
5.8.2	Conditions.....30
5.8.3	Cycles31
6	Use of software in education32
6.1	Advantages32
6.2	User feedback.....32
7	Opportunities for further development.....34
8	Conclusion.....35
	References.....36

Figures

Figure 1.	Working on the IPA6
Figure 2.	Android Studio.....7
Figure 3.	Run on real devices9
Figure 4.	Turn on USB debugging.....10
Figure 5.	Select Deployment Target.....10
Figure 6.	Activity life cycle11

Figure 7. Using Breakpoint	12
Figure 8. APK menu	14
Figure 9. ViewGroup structure	15
Figure 10. Self-made adapter for ScrollView	16
Figure 11. Blockly code.....	18
Figure 12. Scratch User Interface	19
Figure 13. Code.org	20
Figure 14. User interface	22
Figure 15. Menu	23
Figure 16. Variables	24
Figure 17. Variables 2	24
Figure 18. Operators	25
Figure 19. Cycles.....	26
Figure 20. Conditions	27
Figure 21. Pointer	28
Figure 22. Example program 1	29
Figure 23. Example program 2	30
Figure 24. Example program 3	31

Tables

Table 1. Comparison of programs for similar purposes.....	20
Table 2. Feedback.....	33

1 Introduction

In the world of the development of Information and Communication Technologies (ICT), more and more ICT devices are invented, and their quantity and level of development can be increased day by day. Nowadays, almost everybody has at least one smartphone. These mobiles are nearly as advanced as some computers which can be used more easily since they are portable. Their portability stems from their size, but that small compact size does not benefit users in all areas. One such area is programming. There is hardly any programming environment, i.e. a mobile platform which allows coding when travelling.

In the H course of the studies in Hungary, students can be familiar with different development environments and programming languages within the Programming Languages Course and the most students sympathized Java language. The Android Studio IDE specializes in the development of Android applications and one can also build an app in Java, which was found to be the most ideal for the students. The application was prepared by the author and his friend.

The primary, most important goal of the application is to help and ease programming on a mobile platform. The software tries to eliminate the difficulties and barriers of the coding on the mobile interface with an interactive way and specified programming elements.

The purpose of creating the software is to be useful for people who are still familiarizing themselves with IT and IT work. In the fast-paced world, it is important to be able to spend much time as productively as possible. There are many new downloadable mobile applications every day on the market. Some of them are learning and skills development apps, and the thesis would like to expand the range of these. In the following, the Android Studio development environment is presented, a similar software that inspired the author. Finally, the thesis presents the development of the application as well as its operation and further development possibilities.

2 Basics of Android software development

2.1 Researching technologies

The beginning of research was spent collecting and studying the literature. Most of the example tutorials in textbooks were written to deepen an existing knowledge. The YouTube video sharing portal also was a good opportunity to watch tutorial videos which can be very useful for learning new programming technologies. An upgrade of computers is essential; hence, investigating RAM and SSD is recommended. Once the tools were ready for development and the initial steps were clarified, then the opportunity was given to learn and try out different technologies suitable for mobile development. Many possible technologies are provided, for example: React Native CLI, React Native Expo, Qt, PWA and Xamarin. JavaScript, HTML, C++ and different programming languages are used with these. Some knowledge was gained about these technologies and one technology was selected that the author liked the most. (Zelena 2017)

In the React Native CLI, it is more difficult to install or initialize projects which are made in the command line. This can cause problems for inexperienced people. The React language and the Node.js software system are used by it, which can cause compatibility issues.

The Android Studio IDE was preferred because it was much more talkative, understandable and simpler for the author. The Debug system and error messages are easier to use. The integrated development environment is developed by JetBrains and Google, and many companies use their software. Easier management for version controls, an intelligent code editor, a flexible build system and a fast emulator are provided; therefore, this environment looked much friendlier.

Firstly, the Android Studio environment and the Debug system was learned and plenty of issues and problems were solved by using these pieces of knowledge. Under the development, many emulators or real devices can be used. By watching videos and learning on the websites, enough experiences were collected, and some small games could be prepared, for example, Tic tac toe, Coin Toss, and Drag & Drop. After the IDE and almost all tools were familiarized with, then the plan-making was started, for example, the Frontend style was drawn, and the associated functions were selected.

2.2 Android Studio development environment

Applications can be developed to Android operation system's devices using the official development environment of Android Studio (Figure 1). IntelliJ IDEA serves as a base which is a JAVA IDE (Integrated Development Environment).

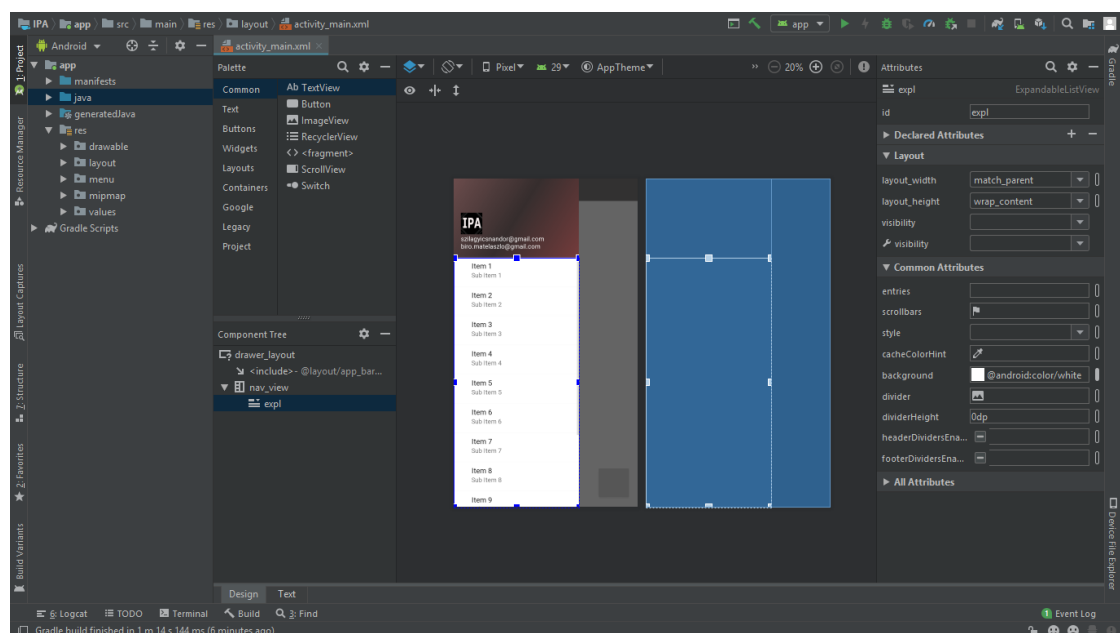


Figure 1. Working on the IPA

Google has introduced new features specifically for Android applications which complement the IntelliJ code editor. Google APP engine was built in the IDE, and it has placed features separately within the project for ease of use; which is why the development and testing process was significantly accelerated (Sting 2017). When the GUI (graphical user interface) was created and edited, the preview and the graphical editor could be helped a great deal in the designing phase. When this application was being made, the Android Studio's version was the newest 3.4.1 version shown in Figure 2 (Hlács 2019).

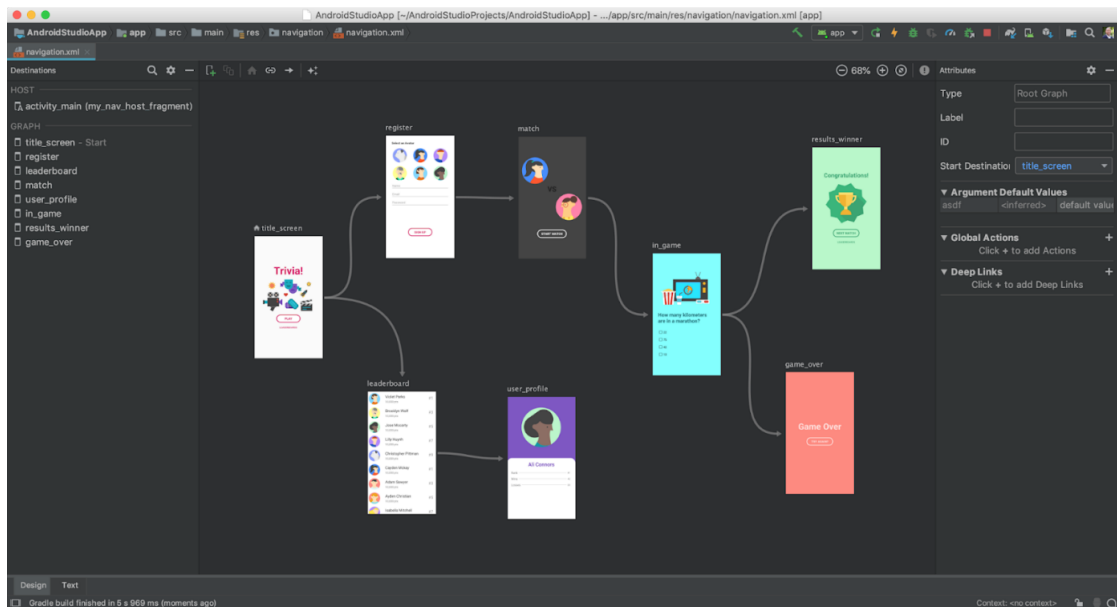


Figure 2. Android Studio

2.3 Test environment

The Android Studio offers an opportunity to run the project in an emulator and in the real device. The ideal environment can be created on the virtual device in which the result can sometimes be unreal. For example, clicking on the screen is often misleading thanks to difference between the screen's cursor and finger. The cursor can be used with more precision than the finger. In the initial stage, Nexus 5.1.1 Android

version emulator was used; because of the above, it has been switched to physical devices, which made the testing easier and faster. Thus, fewer system requirements were required, and it was more reliable.

2.4 Run on real devices

The runnable application can be run directly from Android Studio on real devices. In this case, the device has to be connected to the computer with a USB cable which pairs automatically. If the device cannot be detected, the USB driver software will be found in the USB directory in the SDK folder. Debug can be run on the code, and its setup for the using should be read in the following.

On the device, the information of the system and Android OS can be found under "About the phone" menu which is usually the last option in the settings and can be seen in Figure 3. This menu is available via the "Software Information" which can provide further information about the operating system of the phone. To use it as a successful development tool, one needs to activate the developer mode. To do this, one should select the "Build number" menu item many times in succession until the message "Developer mode active" appears. After these steps, the "Developer Settings" menu option will be available in the Settings.

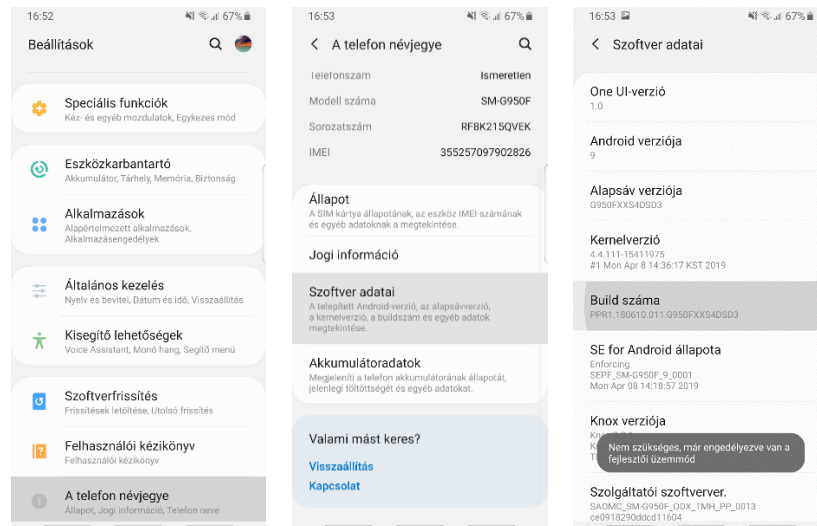


Figure 3. Run on real devices

If the installation file of the application is transferred to the device by file-transfer and then installed from here, it will be necessary to approve the “installation of applications from unofficial sources”. The “USB Debugging” option is important to be enabled, by which the device is able to connect via the USB port (see Figure 4). After the USB connecting, the installations of various interfaces are downloaded. For Android Studio to recognize the phone, the ADB Interface must be successfully installed. After successful setup and connection, a pop-up window will appear on the mobile which asks to enable the USB debugging for this computer. If everything is done correctly, “USB debugger connected” message will be shown on the notification label of the mobile. After pressing the “Run” button on the Android Studio, the virtual or physical device can be selected to run the app on a window panel.

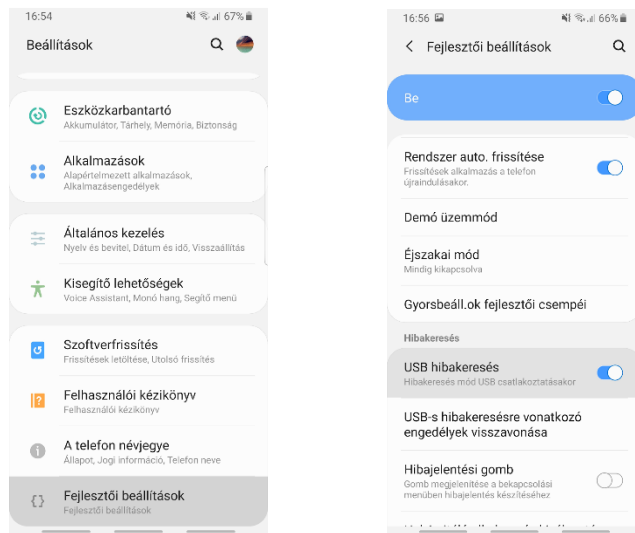


Figure 4. Turn on USB debugging

If the physical device has been selected, the application will be installed first and will start. In this case, the app will be stored on the mobile device after the running. So, to run the already installed app is possible without the help of a computer (see Figure 5).

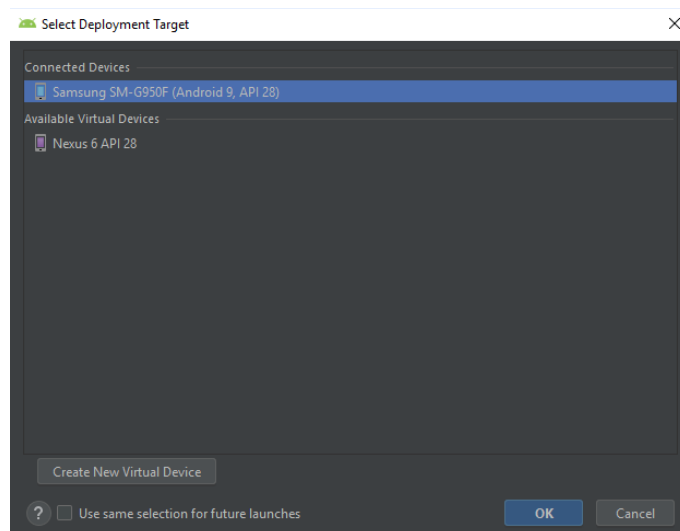


Figure 5. Select Deployment Target

2.5 Elements of Android application

2.5.1 Activity

Activities are the fundamental elements of an application allowing a user to create interactions to access functions. The activities are loosely linked together and an application can contain multiple activities. The activities can rotate the screen or can appear above another Activity. When an Activity is started, firstly the actual Activity will be put into the background, then another can be shown. The activity can be stopped or finished if the device's back button is pushed or the stop method is run. Then the next Activity from background queue is returned to the foreground. The Launcher application comes to the fore when the last Activity is activated in the system. The developer selects which Activity should start first when the application is launched, then the Launcher would be added to that Activity. This is a classic LIFO (Last In, First Out) series. The life of the Activity is similar to the life of the processes in the Operating System shown in Figure 6 (Park 2017).

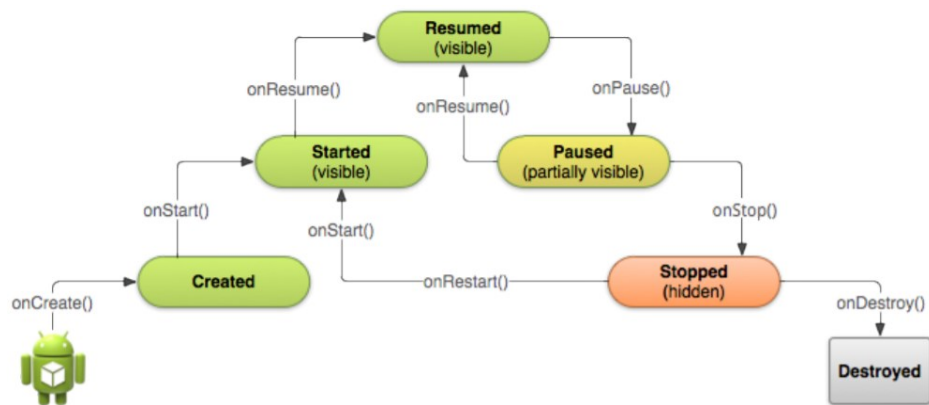


Figure 6. Activity life cycle

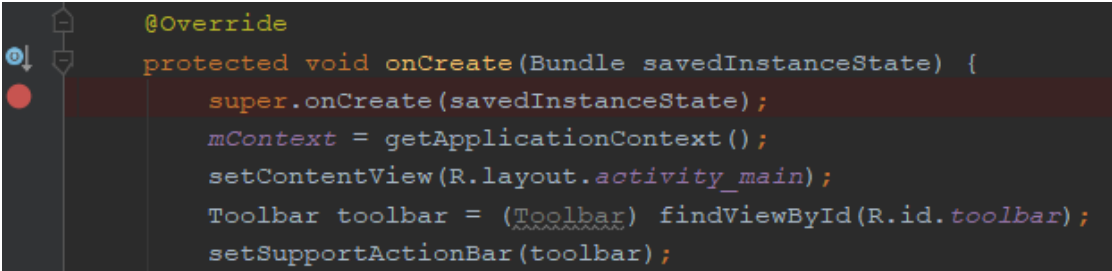
2.5.2 Fragment

Fragments are parts of an application's user interface or an Activity. Each Fragment has its own lifecycle and method. An Activity can contain multiple Fragments and a Fragment can be part of multiple Activities. Fragments are actually modular parts of an Activity. Resource reduction and dynamization can be achieved by using Fragments.

3 Useful accessories for the developing

3.1 Debug running application

While the program is run, an opportunity to get detailed information is provided by Debug run mode. The detailed properties and values of variables and objects are displayed, as long as the given breakpoint is placed. Breakpoints can be placed to the left of the program bar by clicking and removed simply by clicking again. The line is tested where the red circle can be found, and the line highlights in the image. The running is paused on the breakpoint and can be continued in two ways: firstly, by proceeding point by point, line by line, which is also examined, by the “step over” opportunity. Secondly, the running can be continued by proceeding from the actual to the next breakpoint and the running is paused by the “step into” opportunity as seen in Figure 7.



```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    mContext = getApplicationContext();
    setContentView(R.layout.activity_main);
    Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
    setSupportActionBar(toolbar);
}
```

Figure 7. Using Breakpoint

When a breakpoint is reached during the debug run by the application control, the app running is suspended, and another view appears in Android Studio where the contents of variables can be analyzed in a structured way. The current value of the variable (valid when the breakpoint is reached) is also displayed in the source code in the code editor window by hovering the mouse cursor over a variable.

3.2 Publishing applications

3.2.1 A way to publish applications

The great advantage of phone applications is that they can be easily accessed by almost anyone. There are several options for publishing an application, and the application installation package and developer certificate are essential. Google Play Store is the collection point for the best-known mobile apps in the world, where software can be uploaded by everyone.

3.2.2 Developer Certificate

Developer certificate can be created by the software developer for an Android application, which is protected by a password to identify the programmer. These certificates must be properly protected and stored by everyone, as they may abuse personal rights, so the use of certificates is mandatory. Creating certificates is also included in the IDE which can be found in Figure 8 where Build is selected to generate a Signed APK to start it.

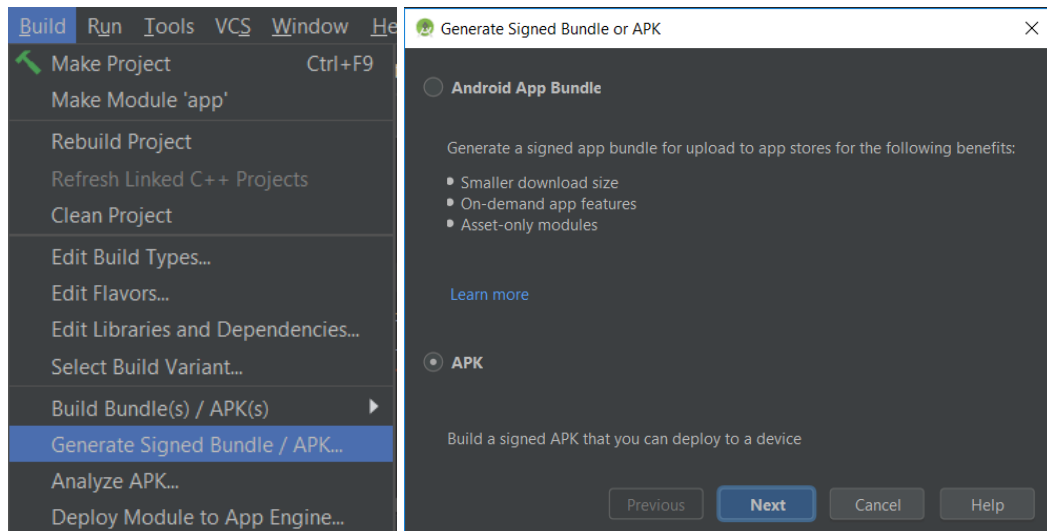


Figure 8. APK menu

3.3 Tools used

The Layouts determine the structure of the application interface. Layouts are hierarchically structured and contain View and ViewGroup objects. View is a part of the display, e.g. a square that contains some elements such as an image, text, button, or anything that an application shows. These separate View elements are brought together by the ViewGroups. The Layouts are formatted in XML, and one root element can be present at a time; hence, if elements are needed for the interface, some unifying element should be used, for example, the ViewGroup. Layout Containers are derived from the ViewGroup class, and it has the responsibility for the interface because Views are included in it as seen in Figure 9. (Divinity 2018)

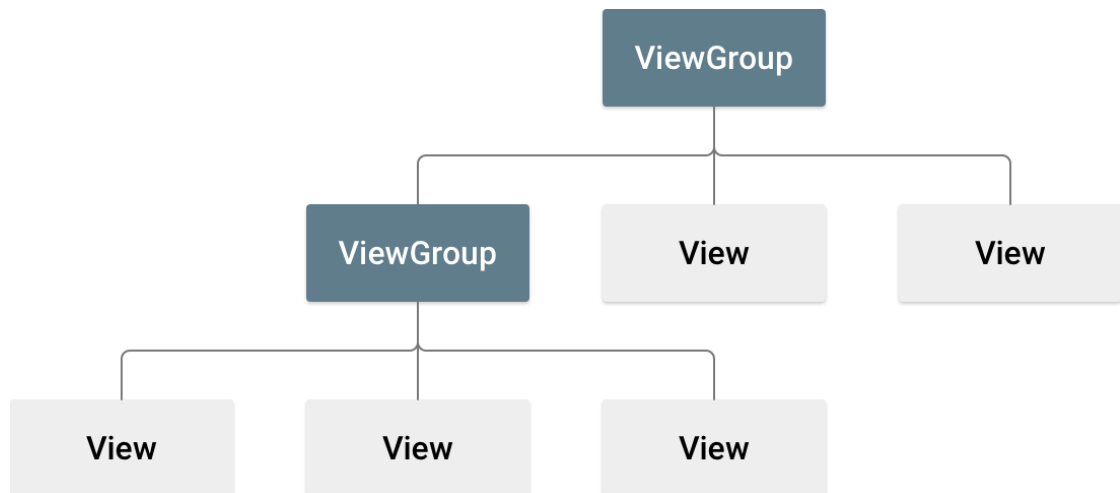


Figure 9. ViewGroup structure

The GUI element is created in two ways, for example, it can be predefined in the XML file or it is dynamically defined at runtime from the program code. The mixture of the two is also used as a common solution. Pre-defined UI elements may be used and they are then modified at runtime.

Common ViewGroups are:

- Linear Layout
- Horizontal Layout
- Relative Layout

The layout can be horizontal or vertical. The point is that the View elements in it follow each other. Its orientation can be easily changed using the `android:orientation` attribute. An important moment is the weighting of Linear Layout, i.e. Layout Weight. The same or a specific portion of the display can be used by each child item of the Layout. Space, which can be gained from the Layout, depends on the Weight of the given element. The bigger screen-space is obtained from the greater weight, and smaller screen-space from less weight. The height is adjusted with Weights, so if the orientation is given vertically, the height should be set to 0 dp. If the orientation is given horizontally, the width must be set to 0 dp.

The Relative Layout is derived from the ViewGroup that is inferred from its name. Its children are placed relative to each other or to the parents. One child can be placed

next to, below, and above the other item without worrying about the order in which the widgets come one after the other.

In the next, the RecyclerView will be briefly described. If a scrollable View is needed, it can be made in two ways. Firstly, the scrollable property can be added to the container by helping of *ScrollView* and the elements are displayed in a *ListView*. In this case, manipulation is made difficult because the entire list is rendered at once, and the use is disabled for a large number of list items; however, it is easy to create this kind of scrollable View as demonstrated in Figure 10.

Secondly, the more difficult but effective solution is RecyclerView. The elements are not rendered at the same time, and it can be solved with its own adapter. For this reason, Java-side manipulation is also required. (Divinity 2018)

```

11
12 public class MyScrollView extends HorizontalScrollView {
13     private boolean enableScrolling = true;
14     public MyScrollView(Context context) { super(context); }
17
18     public MyScrollView(Context context, AttributeSet attrs) { super(context, attrs); }
21
22     public MyScrollView(Context context, AttributeSet attrs, int defStyleAttr) {
23         super(context, attrs, defStyleAttr);
24     }
25
26     @RequiresApi(api = Build.VERSION_CODES.LOLLIPOP)
27     public MyScrollView(Context context, AttributeSet attrs, int defStyleAttr, int defStyleRes) {
28         super(context, attrs, defStyleAttr, defStyleRes);
29     }
30     @Override
31     public boolean onInterceptTouchEvent(MotionEvent ev) {
32
33         if (scrollingEnabled()) {
34             return super.onInterceptTouchEvent(ev);
35         } else {
36             return false;
37         }
38     }
39     @Override
40     public boolean onTouchEvent(MotionEvent ev) {
41         if (scrollingEnabled()) {
42             return super.onTouchEvent(ev);
43         } else {
44             return false;
45         }
46     }
47     @ private boolean scrollingEnabled() { return enableScrolling; }
50     public void setScrolling(boolean enableScrolling) { this.enableScrolling = enableScrolling; }
53 }

```

Figure 10. Self-made adapter for ScrollView

4 Similar applications

4.1 Researching similar opportunities

After the basic idea had emerged, the program was developed. Applications were searched and compared to the imagined plans; some inspiration was gained from them. During this research, the Scratch and Blockly programs were found as well as the Code.org website.

4.2 Introducing Blockly

Blockly is a development platform that can be easily used even by those who do not understand programming. Blockly was not made to be a traditional text IDE, instead of it, it was made to a visual mode. The writing of programs is allowed with matching mosaic elements, which can be easily tried out by children and lay people to whom the basics of programming languages are not known (Strom 2013). This is illustrated in Figure 11 (Blockly 2019). JavaScript, Python, Dart, PHP or Lua scripts can be generated from the visual codes that have been made, thus ensuring the wide applicability of Blockly in applications and related backends. Initially, Blockly could only be run in the browser; nevertheless, it can also be integrated into native applications with the release of 2017, and it is operated on IOS and Android (Divinity 2018). These programs were used to promote the idea because it was shown how to replace different programming elements with interactive elements. Tips were provided, for example, about how to deal with the problem of interconnected items, how to take advantage of and expand the available user interface.

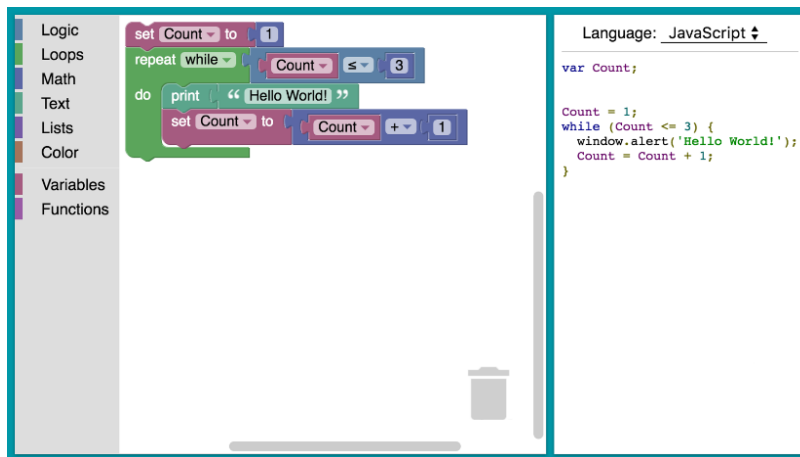


Figure 11. Blockly code

4.3 Introducing Scratch

Scratch is a free programming environment supported by Windows, Mac OS, and Linux that children can learn to program by playing as shown in Figure 12 (Scratch 2019a). The original Scratch is a Squeak-based environment designed for creating games and simulations. It is given in a dynamic interpreted language, so the code can be changed at runtime. The first official version was published in 2007 by the Life-long Kindergarten group at MIT (Massachusetts Institute of Technology; Scratch 2019b). Children's logical thinking is developed, and this will be awarded when learning the basics of programming later on. Simple, playful and funny animations can be created that also develop creativity. The program is made easy to use, easy to understand and visualize with the interactive elements (Knuckles 2000). The popularity is further enhanced by the community experience, as the completed programs/animations can be shared. In Scratch, commands are made up of terms used every day, so even those who are inexperienced in programming can easily get practice. The commands are placed in different colored forms, and only matching elements can be placed one after the other, thus eliminating syntax errors. This thinking is typically promoted so that children can rather focus on solving a specific task and not get distracted by the peculiarities and operation of programming. Nowadays, Scratch has

become available in more and more languages, which increases its popularity due to this, and it is already taught/used in several Hungarian schools.

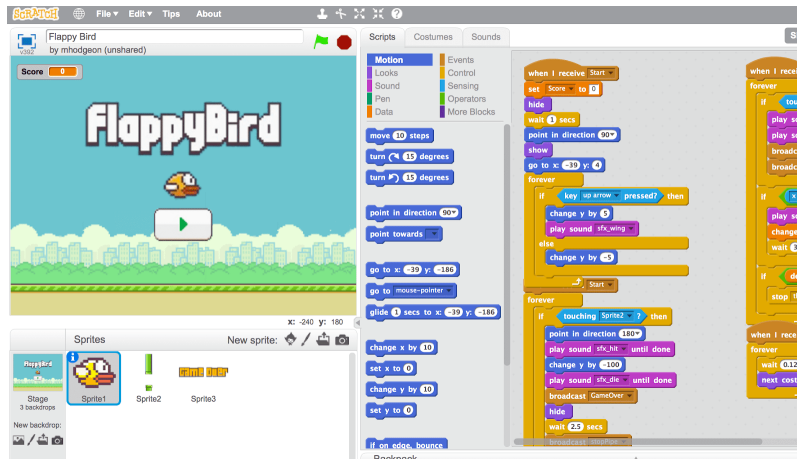


Figure 12. Scratch User Interface

4.4 Introducing Code.org

Code.org is a non-profit organization, and a website was made to encourage people and students to learn computer skills (See Figure 13). On the website, free coding lessons are offered, and the schools are targeted with this initiative to encourage more computer lessons to be included in the syllabus. The site contains tasks to be solved and easily completed by children as young as over 7-8 years old. The mysteries of programming are introduced step by step on the site. The contents are usually short puzzles, the solution of which requires logical thinking, spatial orientation, and geometric knowledge (Code 2019).

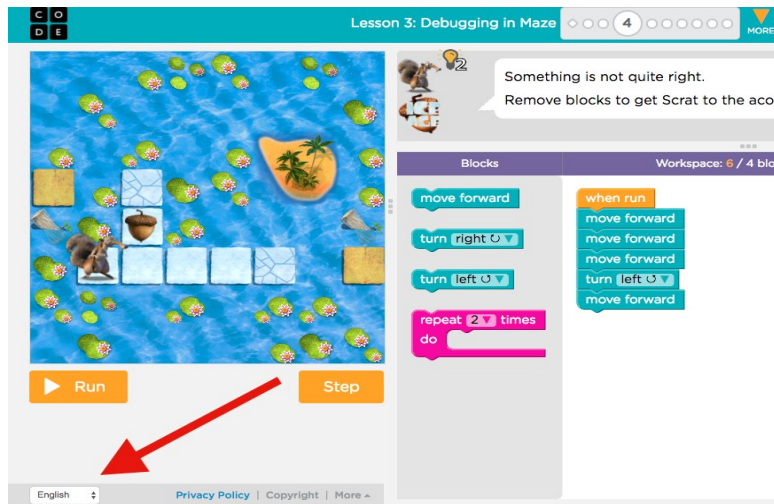


Figure 13. Code.org

The program is created with the help of building blocks, which run the animation of the task. When using code.org, basic programming concepts and methods are shown to the user. (Hírmagazin 2015)

4.5 Comparison

The following results were obtained when comparing the programs. They are presented in Table 1:

Table 1. Comparison of programs for similar purposes

	IPA	Scratch	Blockly	Code.org
High-level	✓	✓	✓	✓
Easy to use	✓	✓	✓	✓
Interactive elements	✓	✓	✓	✓
Animation	X	✓	X	✓
Code generating	X	X	✓	X
Mobile platform	✓	X	✓	X

All four software try to rely on high-level languages. For lay people to use it, ease of use and transparency are essential for these applications. The introduction of interactive elements is closely linked to the ease of use, and the programming process is sped up and simplified. The animation is only available in Scratch and at Code.org, which can mainly increase and maintain the interest of children. The advantage of Blockly is that it can map an interactively created program in predefined languages. IPA and Blockly are available on mobile platforms but there are already versions of Scratch that can be run on mobile devices, and Code.org can be used on a mobile browser.

5 Presentation of application

5.1 User interface

Due to the complexity of the prepared application, it contains many elements and functions, which are explained in more detail below for ease of understanding.

In the upper left corner of the screen, a side menu opening tab can be found when the tab is clicked or moved from left to right (see Figure 14). The Play button is located on the top menu bar, which after it has been pressed, the code will be executed. The line icon or button is also placed on the menu bar which is needed to connect interactive elements or to create a connecting line.

An item can be deleted when it is dropped into the trash icon in the lower right corner of the screen.

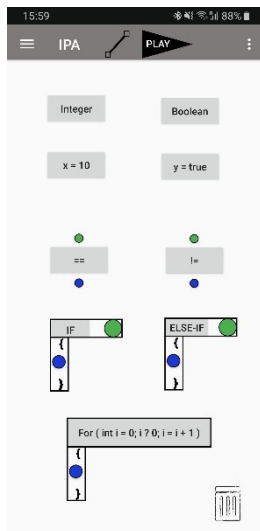


Figure 14. User interface

5.2 Menu Bar

Displaying the menu bar in the upper left corner, the name of the program, IPA, can be seen, which is short for Interactive Programming on Android. The main menu options are located in this side menu bar such as Variables, Arithmetic, Relational, Logical Operators, Cycles, Conditions, Functions. When a main menu option is clicked, a drop-down menu of it appears where the submenu items can be found. The corresponding object / element will be displayed on the screen if one of the submenu items is pressed (see Figure 15).

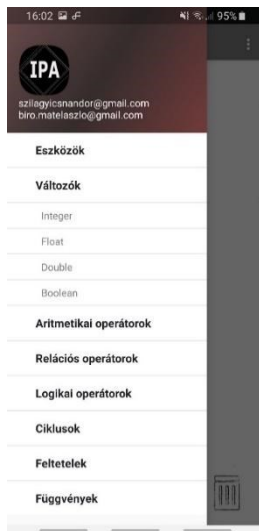


Figure 15. Menu

5.3 Integer and Boolean variables

After pressing a submenu item, its corresponding item is displayed. and the examples of an Integer and Boolean variables are shown in Figure 16. A pop-up window is presented which can be found on the second and third images of Figure 16, if the element is clicked. For the Integer variable, the name can be entered into the first row, while the value can be entered into the second row. In Figure 16, the Boolean's name and value are transmitted, similarly to the Integer, but only here its value can be selected from a drop-down menu.

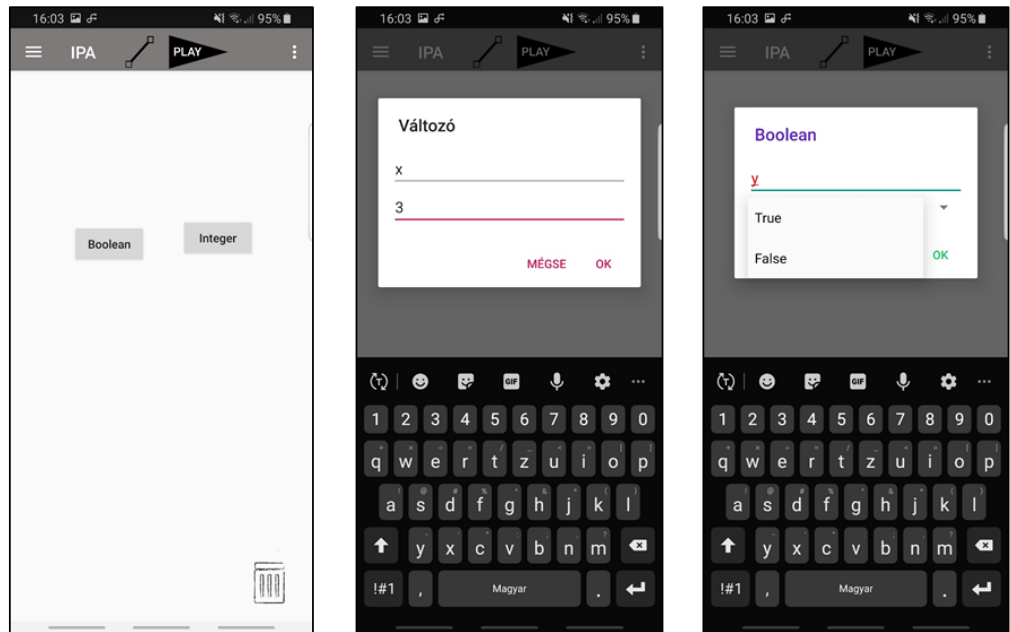


Figure 16. Variables

Figure 17 shows the result of the whole process in an example when the elements are created. The $y=false$ and the $x=3$ variables are prepared in this case.

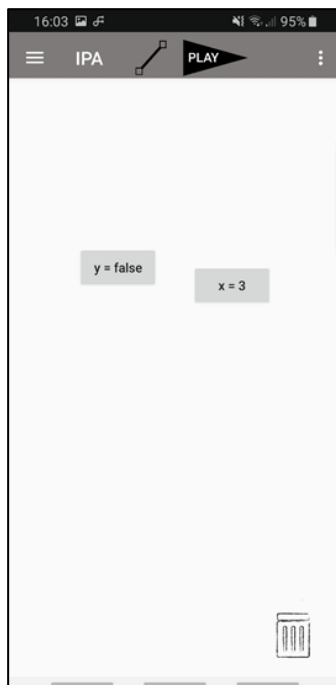


Figure 17. Variables 2

5.4 Operators

Arithmetic, Relational, Logical operators and the associated side menus can be seen in Figure 18. For performing operations, the connections between the interactive elements are prepared by the lines such as the Integer. In the case of Relational and Logical Operators, the input values can be connected to the green circle above. The blue circle below the elements is the output, which is the result of the operation, and it can be used for further operations. So, the result always *true* or *false* and it can be used for conditions or more inputs of complicated operators.

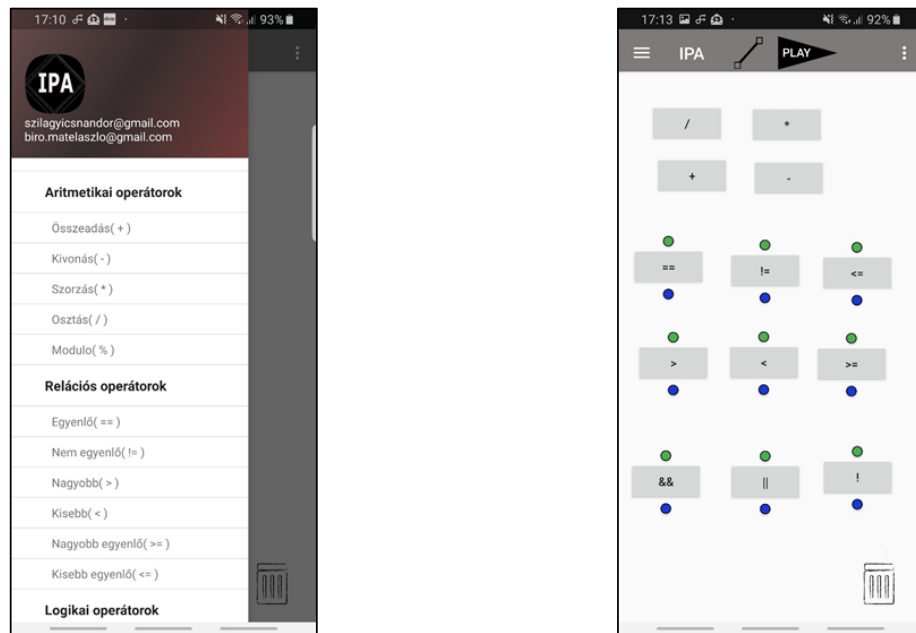


Figure 18. Operators

5.5 Presentation of For, While and Do-While cycles

This chapter presents the cycles. In the first figure, these cycles are displayed in their state after the creation, and the standard state can be seen when nothing is attached to it (Figure 19). After clicking, a pop-up window is displayed for the *for* loop where the name and value of the initialization statement, the relational operator and end-value of the test expression to run for how long, and the value of shifting / upgrade statement can be specified. The relational operator is selected from a drop-down menu, then the values are set by pressing OK button. The green circle of the *while* and *do-while* cycles waits *Boolean* types, which can be *true* or *false*. Instructions can be connected to the blue circle, where the instructions will be run in order, from top to bottom. These blue circles increase downwards dynamically when a line is connected to one circle and the Play button is pressed.

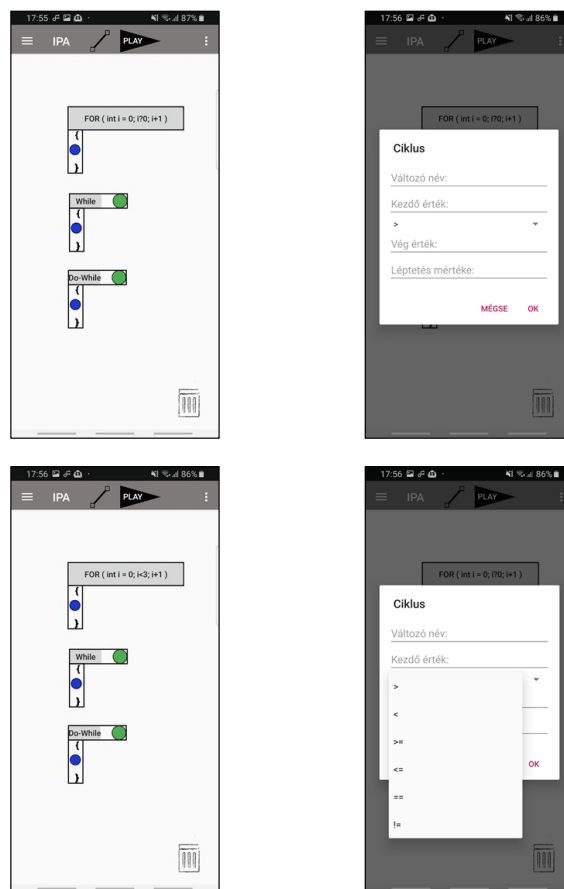


Figure 19. Cycles

5.6 Conditions

IF, ELSE and ELSE-IF created conditions can be found in Figure 20. The *true/false* logical values of Boolean types can be connected into the green circle. The blue circles increase downwards dynamically when a line is connected to one circle and the Play button is pressed.

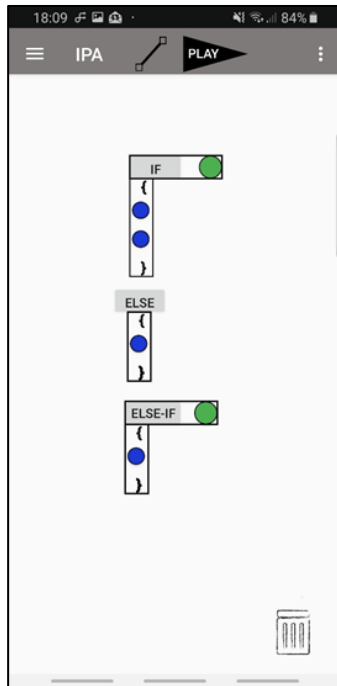


Figure 20. Conditions

5.7 Reference and Value Transfer

Under the Tools menu option, the pointer and value transfer can be found. In the first line, the Integers are created, in the second line the pointers are created, which actually looks like an empty button after the creation in the standard state. In a pop-up window, existing variables can be selected from the drop-down menu by the name as shown in Figure 21.

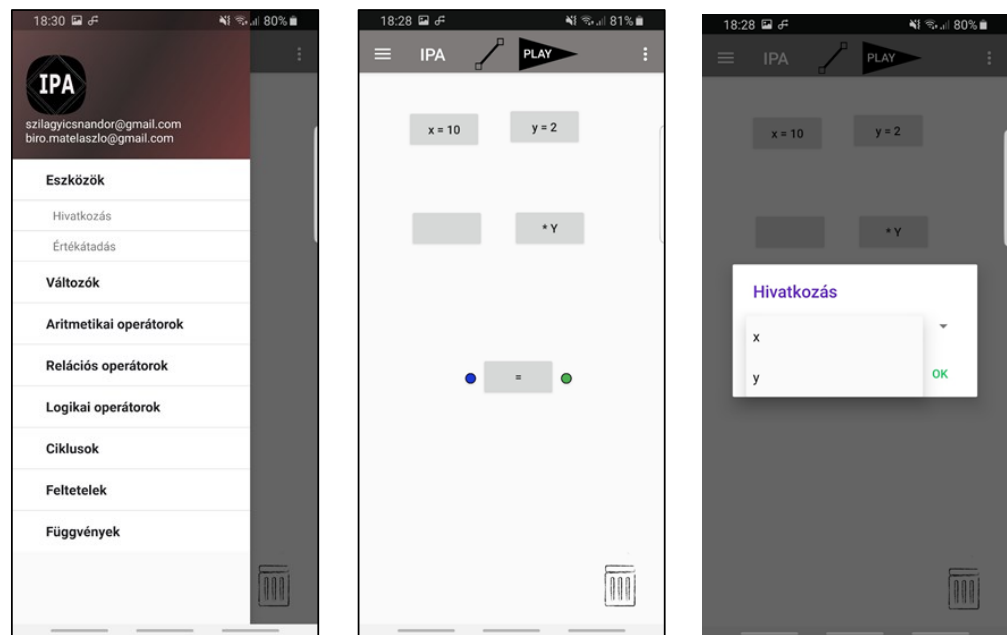


Figure 21. Pointer

5.8 Example Programs

5.8.1 Variables and operators

The following chapter shows the operation of the application with the help of a few example programs. In the first image in Figure 22, a 10-valued variable named x and a 2-valued variable named y were created. Two pointers were also created, and they are referred to as the x and the other as the y . In the second image, the interactive elements are connected accordingly to make the x pointer equal to the sum of the values of the y Integer and the x pointer. After pressing the PLAY button, the value transfer takes place in the third figure. A plus sign before 12 is referred to as the additional element and the 12 is the result of the operation. It can be also seen that the original Integer variable in the first row has also changed.

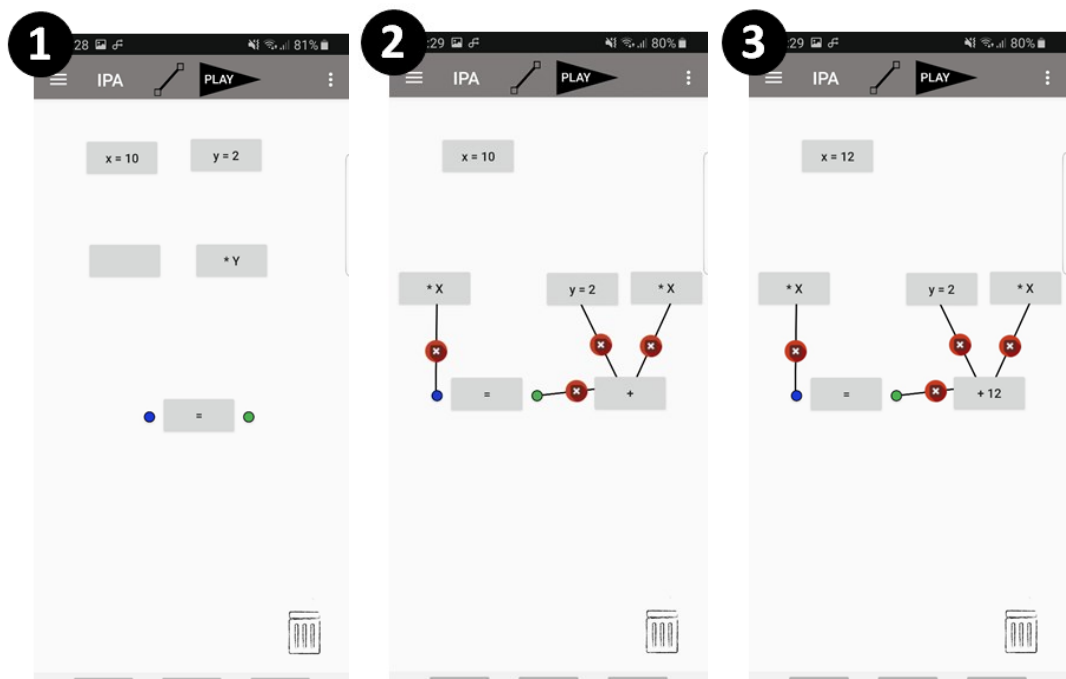


Figure 22. Example program 1

5.8.2 Conditions

The following example program presents how “*if else*” works. In the first image in Figure 23, the appropriate elements for illustration were created. The value of the relational operation for *if* is evaluated and a logical value is returned. In this case it is executed as false, so the *else* connection is run. After pressing the play button, the x - y operation will be performed on the second image. In the third image, the *if* branch will be run, so x + y summation will be calculated.

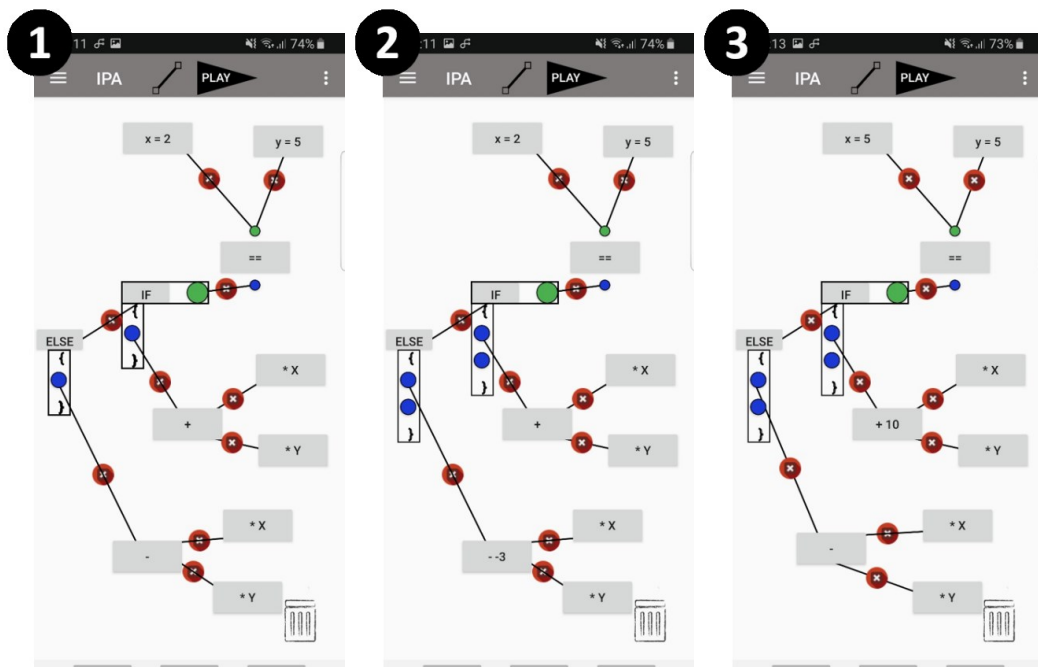


Figure 23. Example program 2

5.8.3 Cycles

In the following simple little example program, the basic operation of the *for* loop is demonstrated. A variable of type $y = 3$ integer is created. In the first image of Figure 24, the *for* loop will be run from $i = 0$ to $i < 3$. Shifting of increment is set 1. After pressing the play button, the value of adding x and y is added to y 3 times in a row, which is solved with pointers. In the second image, the result of 18 can be seen. In the third image, the shifting of increment is just changed to 3, and in this case the result will be 23.

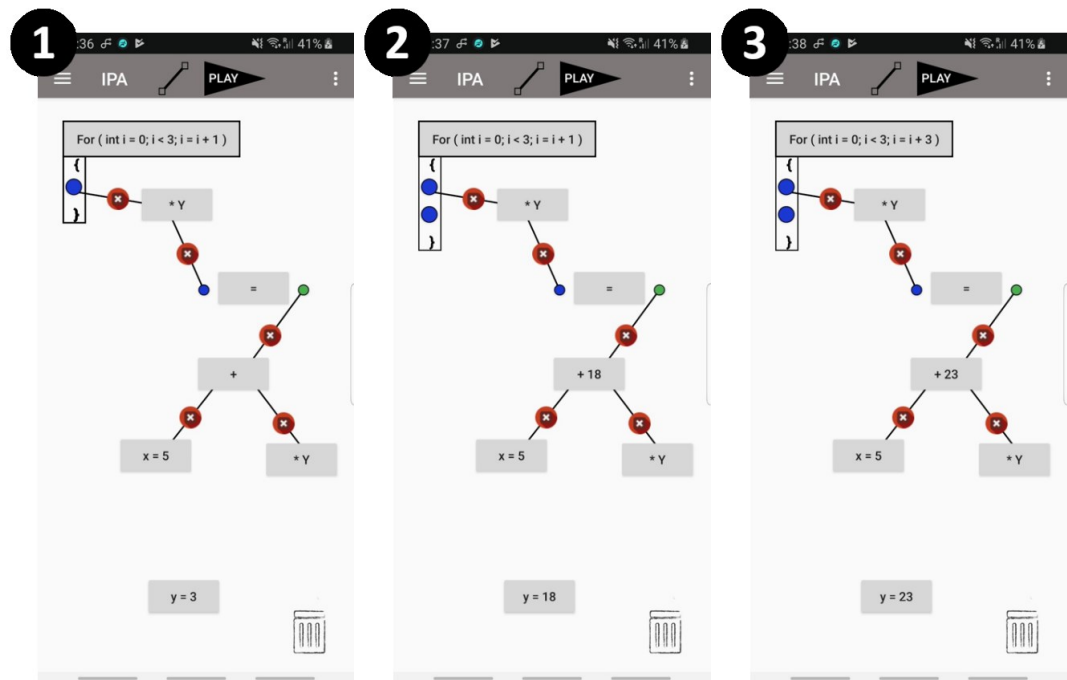


Figure 24. Example program 3

6 Use of software in education

6.1 Advantages

While making the software our goal was to learn programming and to speed up the learning process. The most effective teaching methods have been proven in numerous studies in which new information is conveyed visually. The curriculum can be learnt easily with symbols, figures, different colours and highlighting by students (Ká-tai & Tóth 2010; Shams & Seitz 2008). In this case, when the app is used, the user meets figures and symbols. Thanks to the program's interactivity, it can be put to the service of teaching. The elements can be moved quickly and easily by using the fingers, and a built-in keyboard is hardly needed since a few variable names or values are entered in a writing program. With the interactive content of the IPA, much is included which is more eye-catching for learners than traditional programming methods.

6.2 User feedback

A small research was conducted involving the IT students and some people who were not really familiar with the world of IT (N=20 persons), and the framework examined the usability of the application. The result is summarized in Table 2.

Table 2. Feedback

Viewpoints	Lay people	IT people
Transparency, manageability	Easy, fast	Simple
Simpler programming elements	An explanation is needed	Understandable, clear
More complex programming elements	An explanation is needed	A brief explanation is needed
Solving basic tasks	Can be solved after understanding	Easy to accomplish

In each case, users could create new objects easily and simply; furthermore, creating connections between said objects did not cause any issues. Lay people needed a quick explanation of the simpler elements such as variables and operators, and later a detailed description of the more complex elements was provided. For competent individuals, only the use of conditions and cycles caused some issues; however, after a few helpful sentences, they were able to apply them on their own. For the test, the participants were instructed to create small samples, which were presented earlier. Some help was still needed for inexperienced people in carrying out the tasks and all tasks were successfully completed by people familiar with IT.

Thus, it is believed that the application will be useful in the development of algorithmic thinking and the practice of programming, both for laypeople and IT professionals.

7 Opportunities for further development

The development of the program is far from over. During the tests, much feedback was received from acquaintances, how and in what way it could be improved. The codes mapping function is one example of such options in which the code is generated with interactive elements, and it would be mapped to a raw executable program. It is planned to introduce additional elements belonging to the main menu items of Variables, Functions, to implement the character management interactive object, arrays and more complex functions for faster coding. The functionality of the tool implementing the connection between the elements is to be extended, which makes the created programs more editable and transparent. Also, it would be great if its use became more understandable, and this could be gained by an incorporating Help Center. The implementation of a helper system that can be turned on and off would make the programming more understandable with interactive explanations such as videos or animation. Documentation for interactive objects can be created (Variables, Operators, Cycles) that would also provide a link to documentation for different languages. Novice programmers would be helped with various example tasks, which make it easier to understand how the app works and uses it. After completing the tasks, the user will not even notice how much programming knowledge they have gained, which will make it easier to understand other languages.

Maybe there is the possibility of dividing the program into several versions according to the knowledge of the given user in the field of programming. More sophisticated tools would be avoided with the lay persons and the harder version would be used by students with more knowledge and who want to feel challenged.

8 Conclusion

Overall, it can be said that the most basic programming elements and their functions are included in the application as well it is imagined all in an interactive way in the completed mobile application. A simple, clean look, a clear and easy-to-use menu system have been implemented. Therefore, the application is user-friendly, easy to understand and easier to use compared to “coding” languages. The number type variables and the basic operations have been prepared which are required to perform operators. The conditions and the essential cycles have also been implemented for the executed cyclically instructions. The functionality between the elements can be created via lines. At runtime, the connections are checked, and further actions are performed accordingly. The created programming elements can be used to create smaller programs. The basics needed for programming can be learned in it, and the users are helped to practice and develop in this "world".

In summary, a unique application was coded and the biggest advantage of it its use on the mobile platform. Based on the experience, a very easy-to-learn tool has been developed that assists students to learn a programming language and develop their algorithmic thinking.

With the investment, our knowledge is developed a greatly, and more experiences is gained in both programming and mobile application development. The thesis project has helped to broaden our horizons in the world of programming.

While this thesis was prepared, it occurred to us to continue the development of applications for Android and other platforms in the future, as well as to be more immersed in learning about other mobile technologies and languages. In addition, it would be great to do further research on the effectiveness of using this tool.

References

Blockly 2019. Blockly program. Accessed on 12 May 2019. Retrieved from <https://developers.google.com/blockly>.

Code 2019. Code program. Accessed on 12 June 2019. Retrieved from https://support.code.org/hc/en-us/articles/360000522711-How-can-I-change-the-language-on-Code-org-?mobile_site=true.

Divinity 2018. Android alkalmazásfejlesztés 2. rész: alapozunk [Android application development Part 2: Basics]. Accessed on 12 September 2019. Retrieved from https://logout.hu/cikk/android_alkalmazasfejlesztes_2_resz_alapozunk/android_studio.html.

Hírmagazin 2015. Világméretű összeesküvés: a Code.org titka. [Worldwide conspiracy: the secret of Code.org]. Accessed on 22 October 2019. Retrieved from <https://hirmagazin.sulinet.hu/hu/evilag/codeorg-titka> 2015.

Hlács, F. 2019. Letölthető az Android Studio 3.3 stabil kiadása. [A stable release of Android Studio 3.3 is available for download]. Accessed on 5 July 2019. Retrieved from <https://www.hsw.hu/hirek/59865/google-android-studio-3-3-ide-stabil.html>.

Kátai, Z. & Tóth, L. 2010. Technologically and artistically enhanced multi-sensory computer-programming education. Teaching and Teacher Education.

Knuckles, C. D. 2000. Introduction to Interactive Programming on the Internet. New York City: John Wiley & Sons.

Park, S. 2017. Activity Lifecycle in Android Applications. Accessed on 5 July 2019. Retrieved from <https://medium.com/sketchware/activity-lifecycle-in-android-applications-1b48a7bb584c>.

Scratch 2019a. Scratch program. Accessed on 17 August 2019. Retrieved from <https://www.kiditech.org/courses/game-design-stratch/flappy-bird-in-scratch/>.

Scratch 2019b. A wikipédia Scratch oldala. [Wikipedia's Scratch page]. Accessed on 12 May 2019. Retrieved from <https://hu.wikipedia.org/wiki/Scratch>.

Shams, L. & Seitz, A. R. 2008. Benefits of multisensory learning. Trends in Cognitive Sciences.

Sting 2017. Mobilra is kiadta vizuális programozási nyelvét a Google [Google has released its visual programming language for mobile]. Accessed on 17 September 2019. Retrieved from <https://prog.hu/hirek/4717/mobilra-is-kiadta-vizualis-programozasi-nyelvet-a-google>.

Strom, C. 2013. 3D Game Programming for Kids, Raleigh: Pragmatic Bookshelf.

Zelena, P. G. 2017. "Mobiltudós Guide" az értékesítés segítő alkalmazás fejlesztése ["Mobile Scientist Guide" is a sales assistant application development]. (Bachelor's thesis) Miskolci Egyetem Gépészmérnöki és Informatikai Kar.