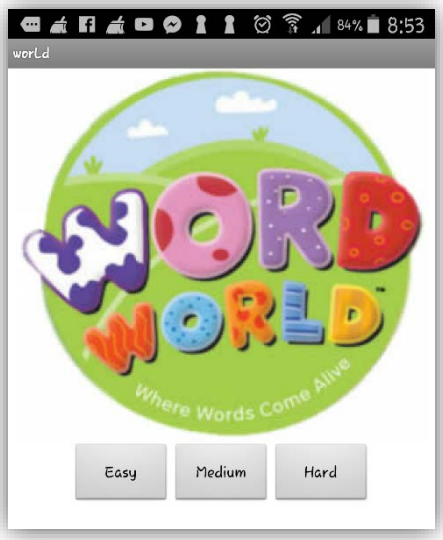
worLd

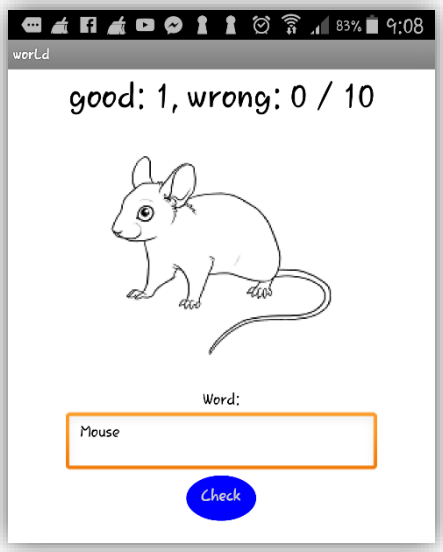
The goal of this application is to help people practice English words and spelling in three different levels (easy, med… etc.)

After you start the programme you will have the opportunity to choose the level which you would like to play.



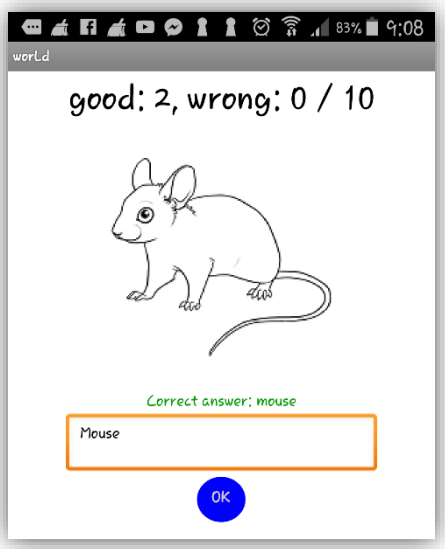
Picture1: Choosing difficulty level

Following that, depends on the chosen level, you will get 10, 20 or 30 pictures. You have to write into the textbox what you can see on the screen (the input is not case sensitive). We write in our tip, after this press on “Check” button to check the word.



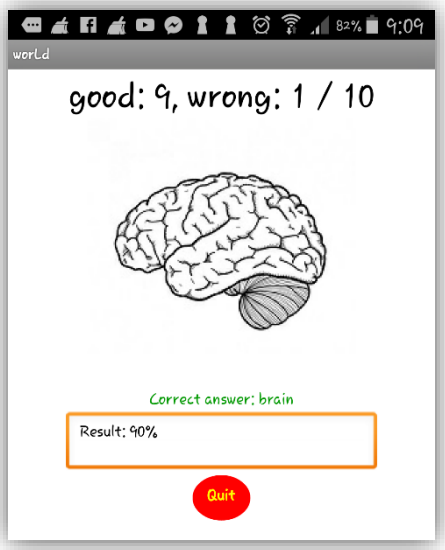
Picture2: Start of checking

If your answer is right, the correct word will be shown in green, in other case will be shown in red colour to help learning the words correctly. On the top of the screen you can see your performance. To confirm that, just press the “OK” button.



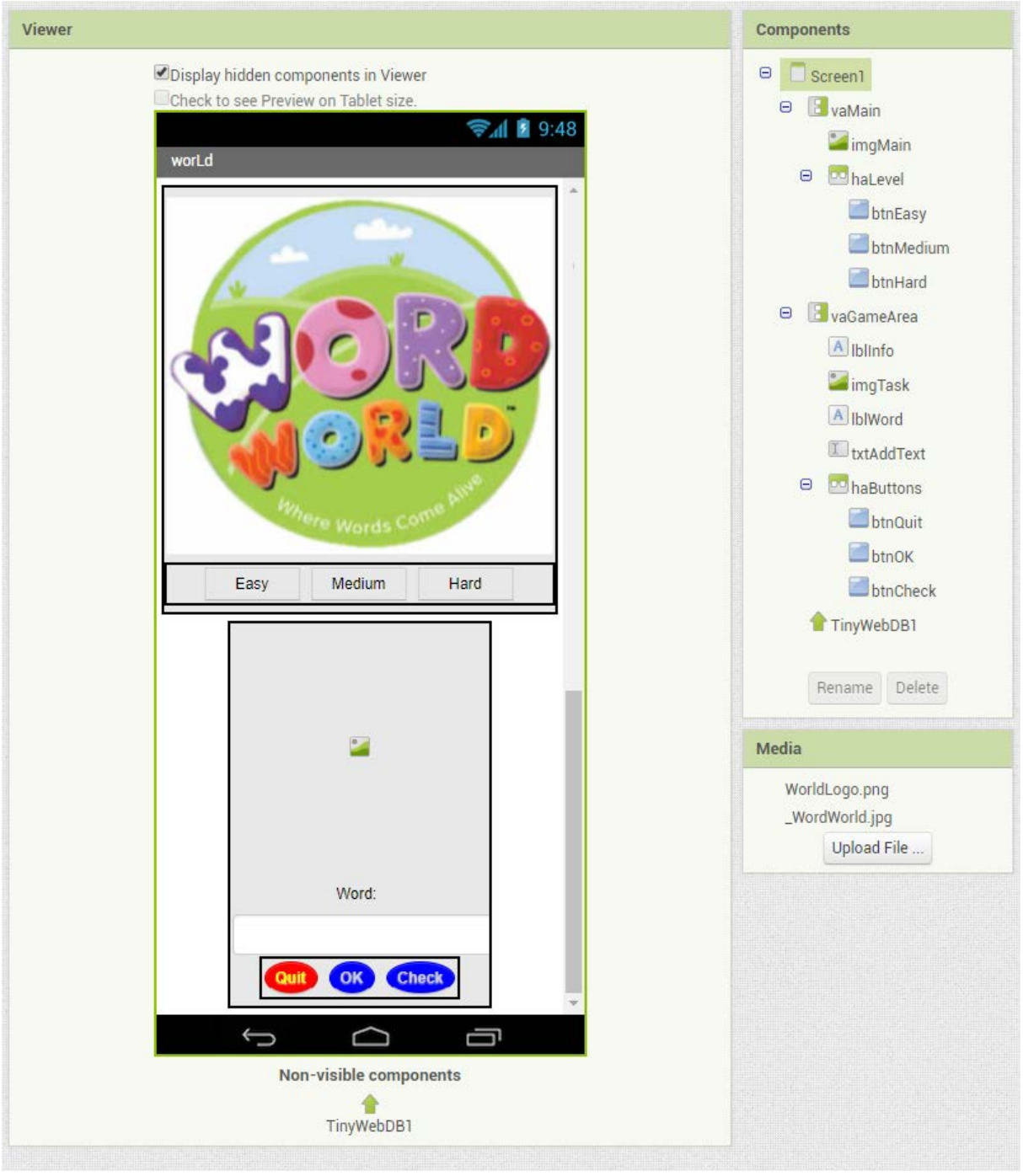
Picture3: Result of checking

After pressing the “OK” button you will get a new picture and word to solve the task. This will repeat until the number of characters indicated after the display / (slash) is reached. Then you will get an evaluation and you can exit from the program.



Picture4: Results and escape

1. Step: Designing the screen



Picture5: Design

Screen: Screen1, Center, Top, Icon: WoldLogo.png, Title: Loading…[1]

[1] It is shown only during loading

Media's upload

a. WordLogo.png: the app logo
b. _WordWorld.jpg: the main screen image

The rest of the media have been already uploaded to a public storage, and their contact information are included in the database.

The main screen (which is visible after loading):

VerticalArrangement: vaMain, AlignHorizontal: Center, Visible ☐ [2]
    Image: imgMain, Picture: _WordWorld.jpg
    HorizontalArrangement: haLevel, Center, Center, Width: Fill parent
        Button: btnEasy, Width: 75px, Text: Easy, TextAlignment: Center
        Button: btnMedium, Width: 75px, Text: Easy, TextAlignment: Center
        Button: btnHard, Width: 75px, Text: Easy, TextAlignment: Center

The design of the game area (which is not visible at start up):

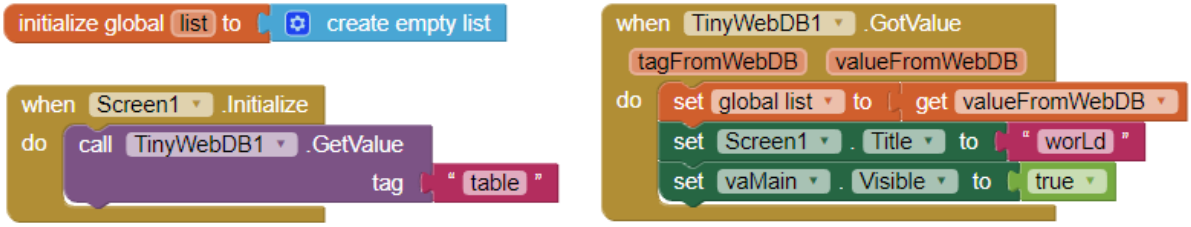
VerticalArrangement: vaGameArea, AlignHorizontal: Center, Visible ☐ [2]
    Label: lblInfo, FontSize: 30, Text: „"
    Image: imgTask, 200px * 200px
    Label: lblWord, FontSize: 14, Text: „Word:"
    TextBox: txtAddText, Enabled ☑ [3], Text: „"
    HorizontalArrangement: haButtons
        Button: btnQuit, BgColor: Red, FontBold ☑, Shape: oval, Text: Quit,
            TextAlignment: Center, TextColor: Yellow, Visible ☐
        Button: btnOK, BgColor: Blue, FontBold ☑, Shape: oval, Text: OK, TextAlignment:
            Center, TextColor: LightGray, Visible ☐
        Button: btnCheck, BgColor: Blue, FontBold ☑, Shape: oval, Text: Check,
            TextAlignment: Center, TextColor: LightGray, Visible ☑

Component not visible:

TinyWebDB1: ServiceURL: https://erasmuspals-157008.appspot.com/

[2] *If you turn off visibility, you can only edit it later when you turn on Display hidden components in Viewer. In all child objects we no longer have to deal with Visible because they inherit the similar property of a parent object.*
[3] *Although this property is turned on by default, it is important to check this setting because you will meet it later.*

2. Step: Scanning the database into the list



Picture6: Database operations

If the database loading was successful, the mainscreen (vaMain) component will be displayed. Here object orientation is used, which means in this case that with setting the parent object (vaMain) to visible the descendants will be visible as well. (see Design)

The structure of the database is special, because on the one hand, a record contains the name of the image and the title of the image with a * (star) separator. On the other hand, the correlation between the 30 records and levels can be seen in the table below:

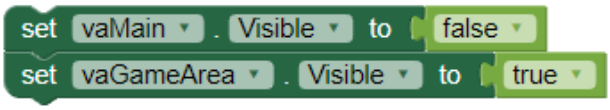
recno	record		Easy	Medium	Hard
1	dog*http://moricz.arrabonus.hu/erasmus/dog.jpg				
2	deer*http://moricz.arrabonus.hu/erasmus/deer.jpg				
3	cube*http://moricz.arrabonus.hu/erasmus/cube.gif				
4	axe*http://moricz.arrabonus.hu/erasmus/axe.jpg				
5	tree*http://moricz.arrabonus.hu/erasmus/tree.jpg				
6	monkey*http://moricz.arrabonus.hu/erasmus/monkey.jpg				
7	brain*http://moricz.arrabonus.hu/erasmus/brain.jpg				
8	wolf*http://moricz.arrabonus.hu/erasmus/wolf.jpg				
9	camera*http://moricz.arrabonus.hu/erasmus/camera.jpg				
10	mouse*http://moricz.arrabonus.hu/erasmus/mouse.png				
11	cupcake*http://moricz.arrabonus.hu/erasmus/cupcake.jpg				
12	ghost*http://moricz.arrabonus.hu/erasmus/ghost.jpg				
13	helicopter*http://moricz.arrabonus.hu/erasmus/helicopter.png				
14	mushroom*http://moricz.arrabonus.hu/erasmus/mushroom.jpg				
15	owl*http://moricz.arrabonus.hu/erasmus/owl.gif				
16	parrot*http://moricz.arrabonus.hu/erasmus/parrot.jpg				
17	postman*http://moricz.arrabonus.hu/erasmus/postman.jpg				
18	rose*http://moricz.arrabonus.hu/erasmus/rose.jpg				
19	soldier*http://moricz.arrabonus.hu/erasmus/soldier.gif				
20	turkey*http://moricz.arrabonus.hu/erasmus/turkey.gif				
21	screwdriver*http://moricz.arrabonus.hu/erasmus/screwdriver.jpg				
22	mosquito*http://moricz.arrabonus.hu/erasmus/mosquito.jpg				
23	fingerprint*http://moricz.arrabonus.hu/erasmus/fingerprint.png				
24	church*http://moricz.arrabonus.hu/erasmus/church.gif				
25	archer*http://moricz.arrabonus.hu/erasmus/archery.png				
26	firefighter*http://moricz.arrabonus.hu/erasmus/firefighter.jpg				
27	butterfly*http://moricz.arrabonus.hu/erasmus/butterfly.jpg				
28	calculator*http://moricz.arrabonus.hu/erasmus/calculator.jpg				
29	lobster*http://moricz.arrabonus.hu/erasmus/lobster.jpg				
30	photographer*http://moricz.arrabonus.hu/erasmus/photographer.png				

Picture7: The database

This means briefly that, for example at medium level, the whole database is read into a list, but after scanning it until the "unnecessary" 21-30th records it will be deleted from the list. This is necessary because we can provide only like this way that the images will follow different orders in each game, to generate a random number between 1 and list_length, and then delete this record from the list after displaying the round. We achieve with this that always a valid listindex will be generated.

3. Step: Setting the difficulty level

You can move on by clicking on the right difficulty level button. Firstly, the main screen (vaMain) will be invisible, then the game area (vaGameArea) will be set to visible. Here we also use object orientation, which means in this case that with setting the parent object (vaMain) to invisible the descendants will be invisible as well.

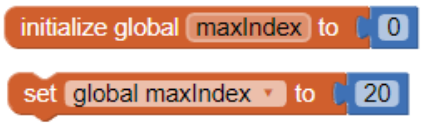


Picture8: Changing Main screen and Game area

After this, for example at medium level, the unnecessary records will be deleted. With a help of a *for* loop records from 21 to 30 will be deleted one by one. In every step the actual first wrong record (21) will be deleted.

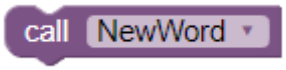
Picture9: Deleting unnecessary records

Then, the number of records will be set (the biggest drawable index).



Picture10: Setting variable maxIndex

Finally, we begin to draw a new word.

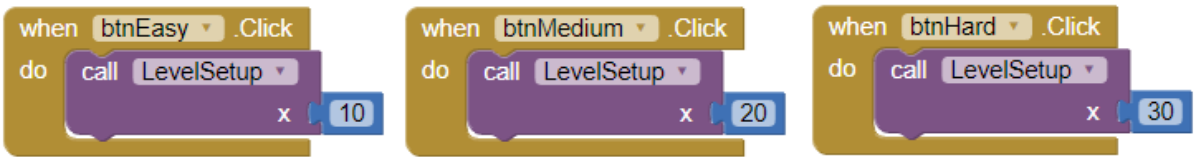


Picture11: Invoking the NewWord event[4]

It is worth thinking about the difference between the operations of the three buttons. After a while, we realize that the only difference is in the numbers of the used records, therefore it is recommended to write a procedure for them, which uses the element number as a parameter. Rewriting the commands above and included into a procedure we will get the followings:

Picture12: Setting the level with buttons

The Click event manager of the three buttons should look like this:

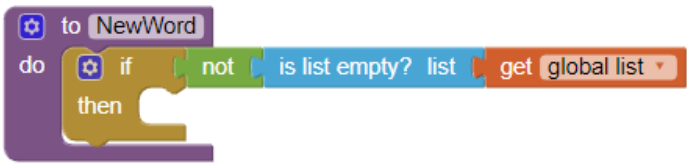


Picture13: The buttons of the level adjustment using the Click events

[4] We are going to use camelcase variables in the followings as well, meaning that the names of the procedures designed by us are also camelcase, but start with capital letters.

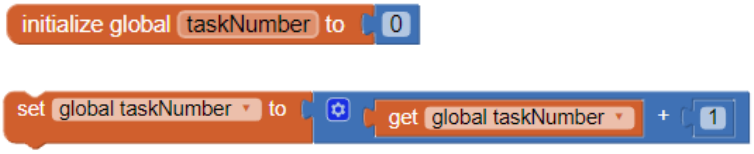
4. Step: Generating a new task from the list

Generating a random word has an important element. It is really important that we analyze the structure of the list and creating a list for the chosen difficulty level (by deleting). We can only generate a new word, if we have an element in our list. Let's suppose our player clicked on Medium difficulty, so the list has 20 elements in it.



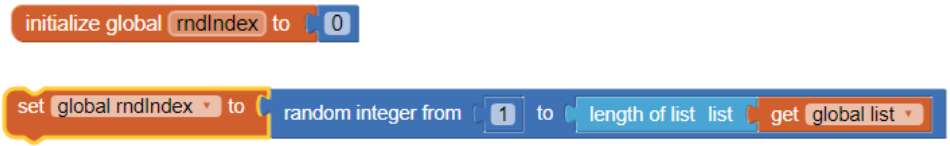
Picture14: The criterion of generating a new word

As the first step of the generating we keep track of the number of tasks we are currently at.



Picture15: Keeping track of the number of the task

As the second step, we generate a random number from the possible record numbers. We are sure that we get a good random number, because we are using built-in blocks. Following this route we are going to get random numbers from 1 to 20.The rndIndex value which was drawn should be 11.



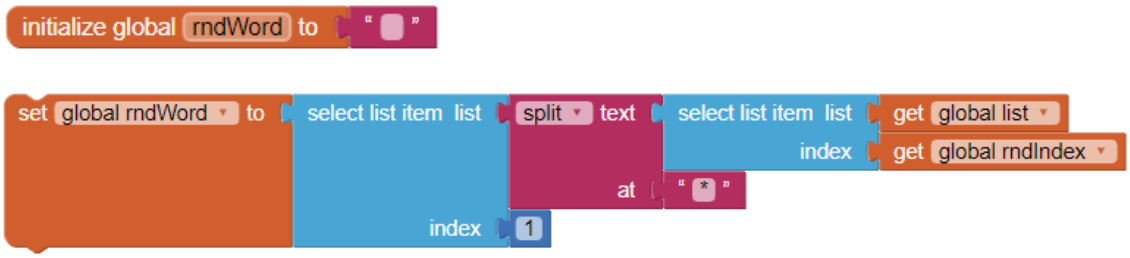
Picture16: Generating Valid random number

From the list we get the item that is attached to the rndIndex and the list contains a "*" character which splits the string into 2 parts. We put the first part into the rndWord. For example the 11th list element is „cupcake*http://moricz.arrabonus.hu/erasmus/cupcake.jpg". A 2 element list is going to be created according to the "*" character split.

[1]: cupcake

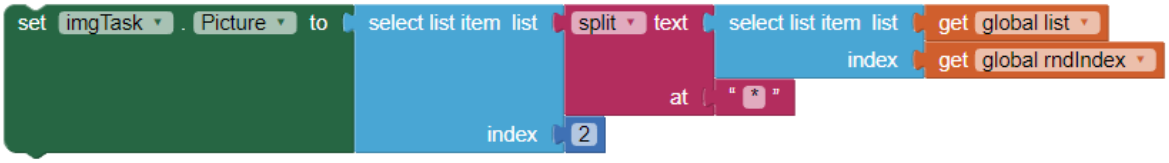
[2]: http://moricz.arrabonus.hu/erasmus/cupcake.jpg

After that, we just have to put the first item of the list into the rndWord variable.



*Picture17: Loading the list item before the selector * into rndWord variable*[5]

The second list item is loaded into the imgTaskPicture property.



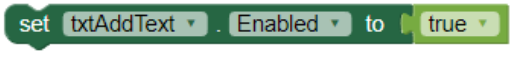
*Picture18: Loading the list item after the selector * into the imgTask Picture property*

Finally, delete the rndIndex item from the list. This works well like this, because without keeping the last list item a new number can be drawn. Moreover, the game will not be tiring because the records will be randomly drawn. So, the 11th list item is going to be deleted.

Picture19: Removing rndIndex list item

Thus, in the next round the *length of list* ***global list*** is going to be only 19, and the random number generator is going to draw between 1 and 19.

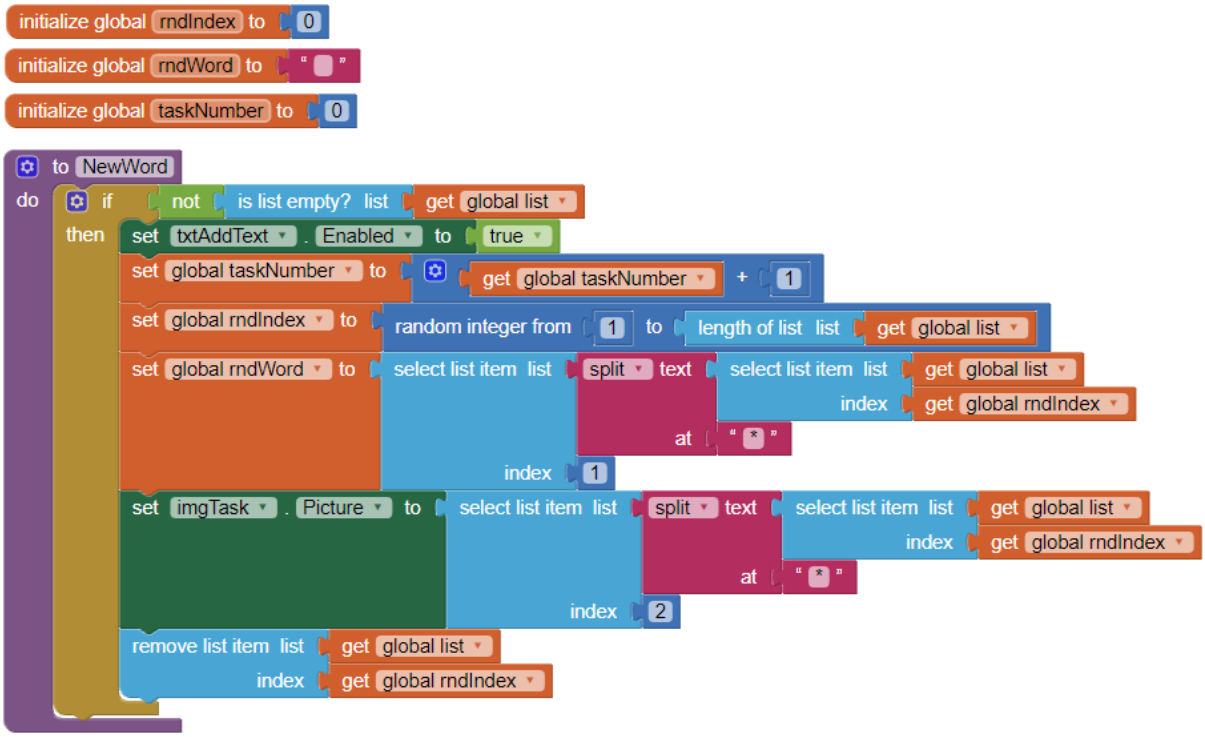
Another line is included in the procedure only because of utility reasons. Since, when the output is displayed we do not want the player to modify the text of the textBox, therefore it is set to disable for that time. And now here it is enabled.



Picture20: Enabling to modify the content of textbox

[5] Split block creates a list from a string as a **text** with splitting by the character (*) after **at**. Working like this the first element of the list will be the word before *.

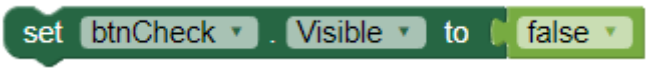
Altogether the block code of generating the new word:



Picture21: The whole NewWord procedure

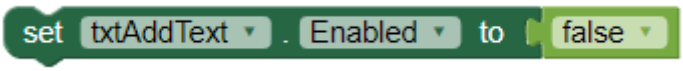
5. Step: Checking the given word

Before we start checking some settings are needed to be done. Turn off the visibility of Check button.



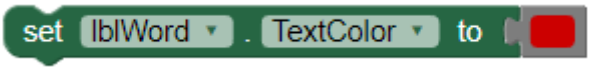
Picture22: Setting the visibility of Check button

Here, we would like to display the results that is why we do not want the player to write into the textBox, so for the time of displaying the results it is going to set to disable.



Picture23: Setting to disable the editing of txtAddText

The default colour of displaying is set to red.



Picture24: Setting the text colour of lblWord to red.

Checking is needed, if the written text and stored text is the same. Setting the written word into not Case sensitive, we need to set the word into written with small letters, the stored words were already saved as written with small letters. If the two of them are the same, the number of scores will be increased and the text colour will set to green.

Picture25: The checking

The results are written:

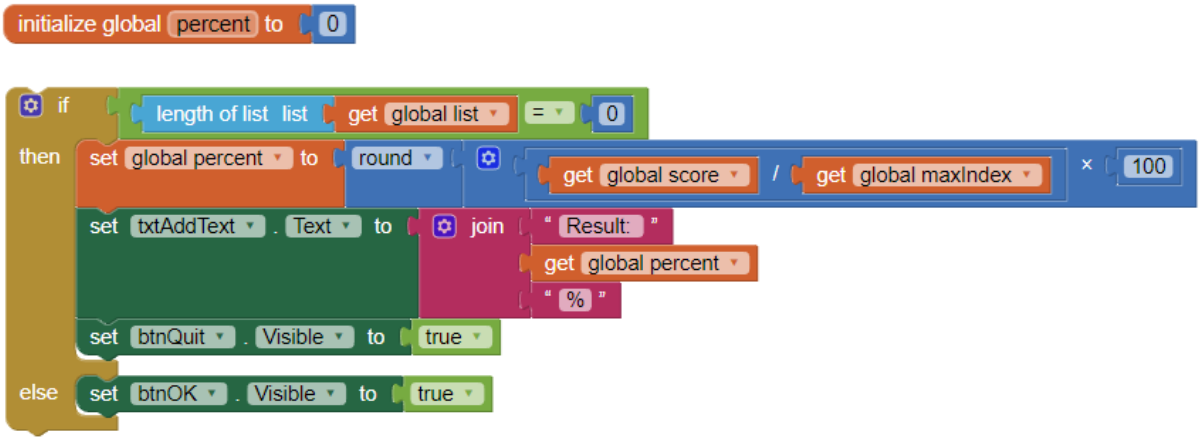
Kép26: Displaying the results

The correct answers are written:

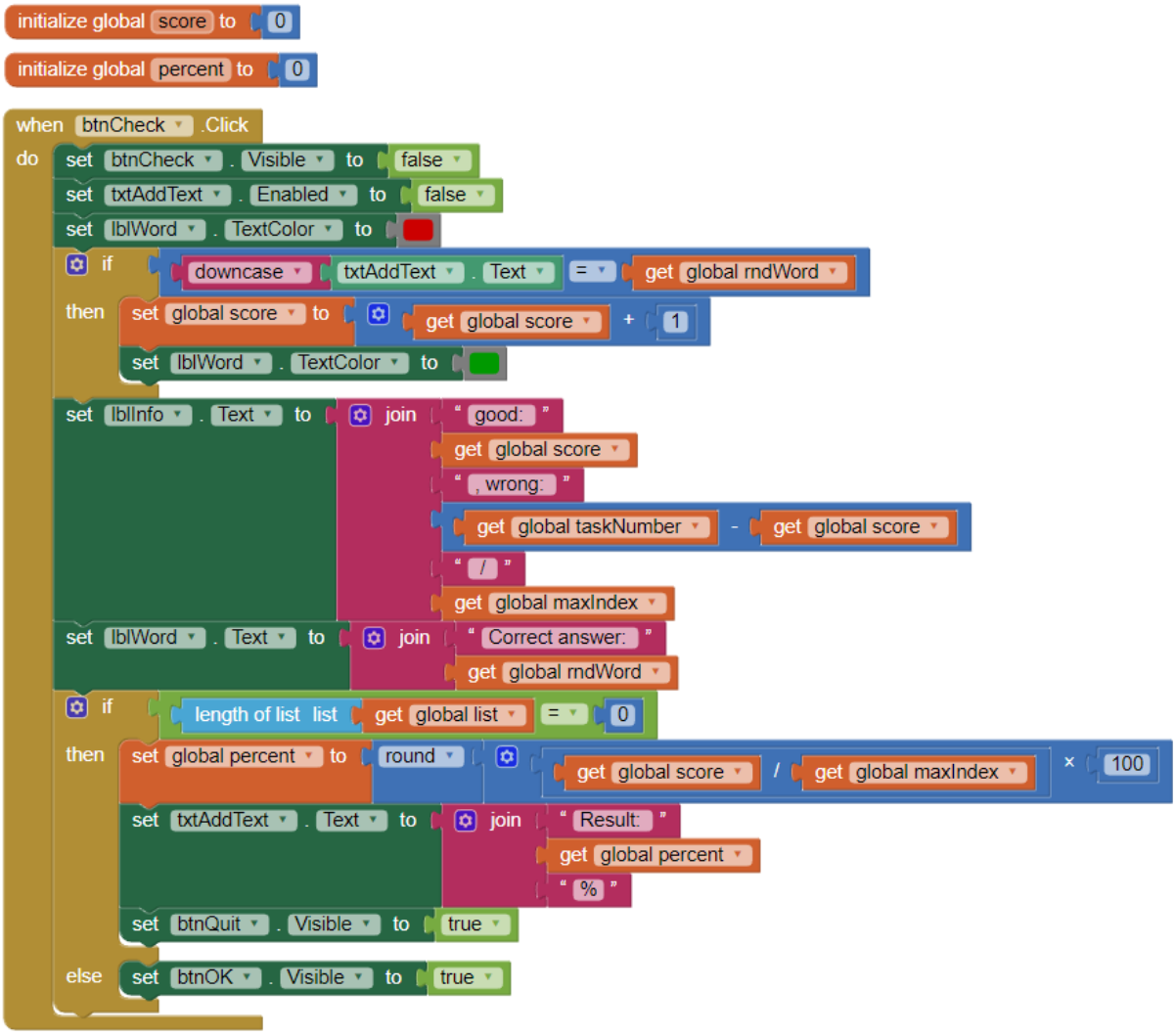
Picture27: Displaying the correct answers

If there is no more list element that means all the tasks had been done. So, an individual result will be count (scores/max score *100). Then it will be rounded to a whole number to be displayed fine on the screen. This result is going to be loaded to txtAddText and set the

visibility of Quit button to enable. If there are any drawable element, only the OK button will be visible.



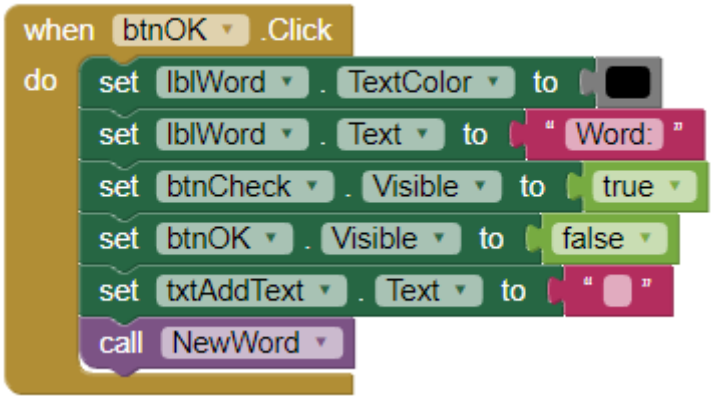
Picture28: If there is no more list item, or there is



Picture29: The full checking event

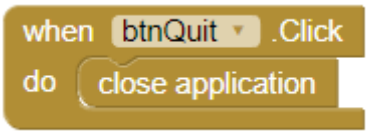
6. Step: Accepting the results and starting to quit

If there is another task to do, then the OK button will appear. For this, some settings are going to be done in the Click event, then the NewWord procedure will be invoked. The settings are the followings: The appropriate answers place is changed by „Word:”, after we changed its colour into black. The visibility of the Check button is turned on, and the visibility of the OK button is off. The input field should be deleted.



Picture30: The Click event of the OK button

The event management of the Quit button is very simple.



Picture31: The Click event of the Quit button

The full block list: