Guidelines

Circle records. Lesson 4

The main concepts discussed at the lesson

- Messages
- Pen
- Lists

Electronic educational resources used at the lesson

- Internet-enabled computer
- preinstalled RobboScratch3, Zoom or Blue Jeans
- Own account in the community on https://scratch.mit.edu/

Features of the lesson's representation

This course has been developed by the ROBBO Methodological Department specially for distance learning.

A special thing about the course is that the material is presented visually in the form of slides. Explanations for each slide follow below.

Greeting.

Hello, children! Welcome back to our RobboClub.

This is our fourth lesson. We will continue to make the Circle vs Squares game. Finished game reference:

https://scratch.mit.edu/projects/394872122 Circle vs Squares

Something was added to it. In the instructions for this version:

Open list: 2, close: 3. Play without changing your name: 1.

Some of you trained at home and fixed your scripting skills.
Open every game you play. Start. Make sure everything works.

Slide №2

Coordinate system. Stages. Name the coordinates of the upper-left and lower-right corners. -240; 180 and 240; -180.

Slide №3

Button controls. There are two different ways to control buttons:

1) Events-Title-When,

2) Conditional operator If the key is pressed.

What is the difference between these methods? Answer to question: option 2 allows the sprite to move diagonally.

Slide №4

Other block. What the Other block and Variable have in common. Answer: Both must be created by assigning a name**.**

What is the Other block? Answer: It is a Subprogramme.

We will continue to develop our game.

Slide №5

Subprogram Deflated (lost). Let me remind you that in the last lesson we agreed on the game rules. Here they are:
1) each touch of the Circle (the Main character of our game) with any moving Square (moving obstacles in the game) increases the size of the Circle by 5%.
2) Another Deflated block is enabled, If the size is greater than 155%. The sprite performs "**Go**" to the specified location. This is a loss.
It is important to remember,

1) The circle can rotate during the game, so it is important to indicate the **direction of 90** degrees. Because of its round shape, turns are not noticeable.
2) You don't need to rotate the Circle if you set the space command "**Rotation style - Don't rotate" at the very beginning of the code.**

Set the commands "**Wait for 3 seconds**" and "**Stop al**l." The necessity of these commands will become clear later. We'll do the unfamiliar "**Pass**" command a little later. This is the end of the Circle sprite programming except for the "**times" Variable**. We will return to it later.

Slide №6

Messages. Events. Now we will make a micro-project to learn how to work with messages. Let's add the Beetle and Crab sprites to the project we are working with. Once we have learned how to work with messages, we will delete these sprites.

Messages are a very useful Scratch feature. Messages are used when one sprite must act in response to an order from another sprite or background.

The "**Send message**" command is located in the **Event** block. There is also an **"Event When I get ..."** the following is a quote from the message. You need to create a message.

Question to the chat. Which of the three options do we create? 1-Timer. 2-Variable. 3. Other blocks. Correct answer: 2 and 3.

Let's look at some of the commands in the Events section. In addition to commands, there are **Headers**, for which the main word (condition check) is "**When**". Scripts using Messages start with Headers that start with the word "When". These Headlines look like caps or hats. For example, this is a Flag we know. We will also need the title "**When the Sprite is Clicked**". Drag it to the script field. Add the "**Send messages**" command to it. Click, but nothing happened. It is because the Message has not reached anyone. Because it wasn't sent to anyone.

Sprites can transmit messages to other sprites. The message must have a name, and the recipient sprite must have a script that starts with the "**When I get**" header (the name of this particular message). There will be a script below that changes the behavior of the recipient sprite.

In the "**Send**" command or in the "**Event When I receive**" click on the white triangle to open the list. Create a "**New Message**". Click it. Write the name of the new message and confirm by clicking OK. The name of the new message appears immediately in the drop-down list of both the Command and the Header. The message you send is the signal for the start of any event. For example, for the "**Say**" event. One sprite transmits to another: light! Another sprite, according to the story of the game (get the Message Light, **Say** Oops!) **Says** Oops! Get scared and pass on your message. So exactly the sprite will be executed.

https://scratch.mit.edu/projects/387341536 **Eye gymnastics (45 seconds)**

Slide № 7

Messaging between sprites. Let's go back to our training sprites. Their programs are on the slide. The script will create a Beetle and a Crab. When you click on the Beetle ...

The plot of the training game Hide and Seek. If you click on a Beetle, it will send a message to the Crab *hide*, and it will hide itself. The Crab will also hide in a short time. Then he'll look out and yell, command **to say** *They're gone!* then it will **send** a **Message** to the Beetle: *They're gone.* The Beetle will get a message, then look out, think again and say *Almost got caught!* We will test and verify these simple scripts within 2-4 minutes. We use this experience in our game.

Slide № 8

Creating a Variable and a Message. Creating the "times " variable. It will change as soon as the Circle **touches** the Finish line. And at the end of the game, this variable will participate in the announcement of the result: "Congratulations" "times", which will correspond to the number of wins. The **Command Set** (variable) "times" **value 0** is included in the script under the green Flag. If this variable is included in the script under the space bar, each start of the game will reset the counter. Our game starts with a Space an unlimited number of times, and the score is reset only from the Flag. After touching the Finish line, the Circle will float to its start. And from there it will transmit the message Victory**!** We will determine where it will be sent. But first we will celebrate this event with cheerful music. Select the sound in your music Library.

Slide № 9

Message sending to 3 addresses. The Circle will send a message about the victory to three recipients at once: Finish, Square, and Stage. The Finish, after receiving the Message, will declare the score of wins. Remember how the string operator is used in this case. The algorithm for the Stage is more complex. Let's take the ready-made Background №2 from the Materials or draw it ourselves. Raster mode, fill with a gradient of two shades, the Text tool, selecting a color, write the word WIN in English. You can write it in Russian. We will arrange a sound track. Background music decorates the game. The selected melody will be turned on by the "**Play sound to the end**" command. It is important to enclose this command in the "**Repeat Always**" bracket. You have already seen the algorithm for smooth music fading with messages WIN and Deflated.

Warm up. **Slide 25** Choose the music.

Slide № 10

Background 2. Musical background. To change the background, it is important to create it. You can draw or take a ready-made one. Try it**!** It develops the imagination**!** We change the background using a message or using the Command "**Switch Background to ... background №...**", and this command is available in any sprite and in the stage. Choose it.

Slide № 11

Message sending to 2 addresses. We have dealt with the victory, now we shoul do the same with the deflation. If the Circle grows too large, it will be broken and deflated. It will send a message to 2 addresses: the Square and the Stage. As a result, the Square will hide. And the Scene will gradually turn down the volume.
The game is done. Test your own game. Do you have any wins?
Improve the end of the game. We will make it more interesting, and at the same time we will get acquainted with another tool.

Slide № 12

Pen. Working with a pen. This is the Pen tool. If the sprite is provided with a pen, it will leave a trace on the scene when moving.

You can see all the commands you need to draw with a Pen on the slide. This program will now be done as an intermediate exercise.

Add the Robbo sprite. After completing the exercise, we will delete this sprite. Attention! When starting a training sprite, do not click the general flag! Click the flag in the sprite script. Is it possible to decorate the beginning of the script with a cover instead of a flag? **When the key (**for example**) 1 is Pressed.** The sprite will launch by the 1 key.

Sprite 1 is our Robbo. We'll let him move.

Flag (or 1) - Go - Turn in the direction - Set the rotation method - Repeat always: (Go - if it touches-Turn to). The last command is just for the prettiness of the line (bonus). This is a sprite movement. Does it work? I hope, it is.

If Robbo had held a pencil in his hand, the pencil would have left a mark on the stage like a piece of paper. This is the main thing in our algorithm. Take the pen first!
Go to the Pen command block. It may need to be found through the extension. When we take a pencil (pen) in the real world, it already has some size and color. We must give these properties to a virtual pencil. Therefore, select the "**Set pen color and Set pen size**" commands. Start drawing on a clean sheet. Therefore, select the "**Erase all**" command - this will clear the stage from the previous drawings. Add the "**Raise pen**" command. This guarantees that the sheet is clean. And once raised, then before you start drawing, you need to "**Lower the pen**". And from that moment, the pen will start drawing with the color and width of the line that we have chosen. Add another bonus: in the Loop operator, insert the command "**Change color to 10**". Click the flag. Admire. Deleting the script.

Slide № 13

Pen. WIN. Sprite performs "**Show Up**" after WIN message!
Long scripts have to be cut to fit on the slide. Arrows show the way to connect the clippings in your mind.
In the Start subprogram contains pen parameters, and the End subprogram contains Robbo animation and congratulations on the announcement of the victory score. Subprograms W, I, and N, in turn, include the Length subprogram, which we will consider later. "**Change x to 22**" after W and I letters, provide a space between the letters. "**Change x to 22**" is the same as "**Go 22 steps In the direction of 90**". But since the sprite was moving at an angle of 15 degrees, the command "**Go 22 steps**" should be set to "**Turn in the direction of 90**" so that the sprite moved horizontally. therefore, it is easier to command "**Change x to 22**". Change x - is always horizontal movement. The y change is vertical. If you put two commands in a row "**Change x to 11 and Change y to 11**", the sprite will move diagonally. The algorithm for writing letters will be discussed later.

Slide № 14

Pen. Beginning Subprogram. It is necessary to set the initial values of the sprite and pen. The sprite size 33% is optimal. Animation will not be displayed when the sprite is moving, so choose "**Costume 1. Rotation Style - do not rotate**". The sprite must reach the initial coordinates without touching the "sheet", i.e. it is necessary to "**Raise the pen**", and everything that was drawn earlier (this happens) – "**Erase everything**". After that, on a blank "sheet", "**Lower the pen, Set the color for the pen, Set the size of the pen**". Now the sprite will leave a trail when it moves.
So, we taught the Sprite how to draw.

Слайд № 15

Pen. End Subprogram. When you finish writing a word, the sprite should move to the final position. We identified it as the top of the letter I. It is necessary to move without leaving a trace, which means "**Raise the pen**". At the top of I, Robbo will dance a victory dance, implemented by changing costumes (**Next costume)** after a pause defined as 0.1 seconds. There are only 4 costumes, each one will be delayed for 0.1 seconds to give way to the next suit. This is how cartoons or animations are made.
Ask students to conduct an experiment-increase or decrease the delay value – "**Wait ... seconds**". Start with 0 seconds. The command "**Wait 0.1 seconds**" can be duplicated by adding the "**Cyclic operator-Repeat 10**", changing the value "**Wait ... seconds**".

After repeating the costume change 11 times, go back to Costume 1. Stopping, the sprite will congratulate and announce the score. You can get a ready-made greeting from the Circle. **Erase, Hide, Stop everything.**

Slide № 16

Pen. Font size subprogram. The Length block defines the length of a straight line segment before changing the direction of the sprite. This is the size of the letter or font. This is a subroutine that is embedded in a subroutine that is embedded in a program. I.e., the length determines all other parameters of the word "WIN". You can change the speed of writing a word. The more **repetitions** and the less "**Go ... steps"**, the slower the word will be drawn. You can also change the size of letters, think how?
We chose the red color of the letters. If you add "**Change the color of the pen by 10**" to the Length, you will get a rainbow.

Slide № 17

Pen. Subprogram letter by letter. The spelling of the letter is determined by the commands "**Turn in the (desired) direction ...**" (degrees). The direction is determined visually by the dial, experimentally. In one direction, the sprite passes a segment defined by the Length subroutine. In the letter W, the line slopes 15 degrees, but the directions are different: 165 and 15 degrees. In the letter N, the angle of the line is twice as large, 30 degrees, so the line is lengthened by an additional command "**Go 6 steps**".
Within a single letter, the sprite draws continuous lines (straight or polylines). And the gap between the letters provides, as already mentioned, the change of x to 22 with the raising of the pen. (Main script, slide 13).
The sprite is ready to execute its program check. In the Materials this sprite is available in a ready-made form.

Warm up. **Slide 25** Выбираем музыку.

Slide № 18

Lists. Continue to complicate our game.

The most valuable thing in the game is the record. To track the best players and identify records, you need lists of participants and records of their results.
Scratch has a convenient interface for creating lists and processing them. In programming, this is called creating, processing, and storing data, arrays of data. The word list itself implies the ability to keep a record. We often write down things, items, and plans in a column. A column consists of rows or records, and the rows are numbered. The number of each row is called an **Index**. Each numbered row stores some data. This data is called string data. The

lines in our list will store data in the form of the name of the participant and the time of his race. We will create a program in which the best players can be identified in the list. Manage a list of sprites.
We will grant the **Finish** sprite the right to mark time and register records.
The list will be located in the Variables command block.
Let's create a list called Results. After creating the list, we will see its monitor. It's still empty. We see 12 commands for managing the list.
Since the list was created to account for records, we will create the **record** variable. Not all of the results are record-breaking. You need to select a record from all the other results, so we will create a **time** variable. You have already worked with this variable.

The list can cover a significant part of the stage, so we will hide it "**When key 2 is pressed**" and show it if necessary with the command "**When key 1 is pressed**". We create these scripts immediately.
On the slide, we see a sample list. It consists of elements or strings. The lines are numbered. The number of the row (element) is called the **Index**. The list knows all its indexes, so you can use it. It also knows its length. Add data to the list of possible commands "**Add to Results**" and "**Insert element ... in the Results on the position with index ...**", the First command adds the line below to the end of the list, the second - by default 1 row, or in any handwritten.

Slide № 19 and Slide №20

Lists. Commands. So, we have created a list. let's get acquainted with its tools. The easiest way to do this is to pull commands to the script field. Click on them and watch the effect of each click. This slide shows these consequences, but only in the example list that you saw on the previous slide. This list is based on the actual results of real races of a ready-made game, but we are only doing this part of the game. so you can fill in the list manually with fictitious names and indicators, as in slide 20. You can see that the list can even be announced by a sprite if the list name is inserted in the command "**Speak ... seconds**". It is important to understand the difference between the "**Insert item to position**" and "**Add item**" commands. It is **Inserted** in the designated index, and **Added** simply to the end of the list. You can see that you can also **Insert** it at the end of the list if the insertion element is 1 more than the length of the list. You can also use this command to change the list items one by one: "**Insert item 3 to the position with index 7**". But here we should not assume that element 3 is line 3 and we will insert it in line 7. Element three is 3. In other words, an element is a record, the line content.

Другие команды. Спрайт может "огласить" весь список посредством команды "**Говорить секунд"**. The reporter can do the same-just click on the title. By clicking on "**List Length**", we will see the length. The Reporter "**List contains?**" enter the word we are interested in and the answer will be Yes or No in the form of true or false which means **true** or **false**. You can clear the list using the "**Delete all"** command. There is a command and selective deletion. We don't need all the teams right now. You will learn the rest when creating your own projects.

Slide № 21

Lists. Start of work. Sprite Finish. We have created lots of variables. The flag will hide the variables and the list. It is important to set the **Record** variable immediately with any large number. This should be a high enough value that it is not difficult to set the First Record. Then the first WIN will replace this number with the real result and add it to the list. And each subsequent race, if it ends in even less time will be entered in the list of records higher than the previous one. The program will start working with the list after the **WIN** message. Losses, i.e. the message Deflated are not interested for the list.

https://scratch.mit.edu/projects/387341536 **Eye gymnastics (45 seconds)**

Slide № 22

Lists. Registration of participants. Our game starts with the Space bar. Let's leave it as it is. But every record has an author. Therefore, by pressing the Space Bar, we will register, i.e. we will introduce ourselves to the judges and they will put us on the list. This is done by using the command in the sensor "**Ask What's your name? and wait**." Let's get to know this important command. When it appears, it is necessary to answer something to the question posed by it. Otherwise, it will not allow you to proceed with the rest of the algorithm. In our game, the Finish is located in the lower right corner and will be a waiting banner for the answer to obscure the banner of the question itself, so that the gamer will not understand what is required of him. So the sprite for the duration of the question "**What's your name?**" and the following parting words, let it go to 0; 0. And the game participant will see the question clearly. He will answer it - give his name, say, John. In response, the program in the form of a sprite Finish will say: Run, John, run! We will provide this with a string operator, using the **built-in response variable** to generate a phrase with your name. After uttering a parting word, the sprite will return to its corner.

Next, the "**Set Record value**" command will be set to "**record**". So far, this is 33, as written under the flag. The following commands: set the variable "**time**" 0, reset the timer. And decorate it all with the message "send start". This message is addressed to the Square and its clones.

Slide № 23

Squares in the game scenario started the chase with a Space. This is unacceptable now: while there is a registration of the participant, the Circle control is not available - you will be blown away. This is why the additional START message was entered. Otherwise, the Square is unchanged. There are no changes for the circle, because it will stand until you start pressing the arrow keys yourself. On the right there is a script that can be added automatically. By clicking on the Space bar, we enter our name each time. This will quickly get boring. After making this script, the name will only need to be called when the Space bar is pressed for the 1st time, and the next races will start with the A key (Latin) - the name is already in the List. This is the final improvement and refinement of the game.

Slide № 24

The script creates a list and organizes its purpose-to identify the record holder. Starts working after receiving "**message WIN!**" 1. Assigns the **timer** value to the **time variable**. 2. **Adds** a new line to the list from the bottom. The line contains the participant's name (received via the **response**) and the race time **combined** with it by the **string operator** (the **timer** value). The content of the line is called an **element**. 3. Then apply the **conditional operator if.** It compares the **time variable** with the **record variable**. If the **time** is less than the **record**, the **record** variable is set to the value of the **time** variable. So the **record** variable decreases. For this, the **element** (the content of the added line) is **inserted at the position with index 1**, i.e. in the top line of the list. However, the entire list remains, which means that its length increases by 1 (the length of the list + the new top line). And now we have the same lines at the top and bottom of the list. This will not change the total number of lines, so we will delete the bottom line. It should be understood that the length of the list is equal to the number of **wins (variable times)**. (**List length**) = **times**. But when we insert an **element** in the top line, we get: (**list length**) = (**times + 1**). This means that you should delete line № (**times+1**) as an extra one. All 3 commands are enclosed in a **conditional operator if.**

We made an exciting game and designed it according to professional standards. It is possible to count, record, and award the winner with animation in this game. What else is missing? There is no separate bonus for the record. There is no initial screen saver, which is appreciated by both consumers and judges of the Olympics. We will not spend more time on this game. Think of it yourself!

Link to the game with all our achievements in programming. In this version, three headers have been replaced.
Open list: 2, close: 3. Play without changing your name: 1.
Instead of 1 it was A in Latin alphabet - switched the keyboard layout once again.

https://scratch.mit.edu/projects/394872122 Circle vs Squares

Slide №25 Choose the music.

Complete tasks in Activity Books.

Task № 1

1 - 5 - 2 and 4 - 3 - 6 The first script makes the second one sing, and the second one can sing alone.

Task № 2

Raise the pen and Erase everything clearly out of place.

Task № 3

Think.

Task № 4

Index.

Task № 5

Talk.