Methodological Guidelines

My First Game. Lesson 2

Basic concepts covered at the lesson

- Timer
- Variables
- Sounds
- String operator

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.

Greetings.

Hello, everyone! Welcome again to our RobboClub.

Today we are going back to our underwater world to continue making our computer game: https://scratch.mit.edu/projects/397006337/

And this is a version of the same game, but with clones

In this lesson we will apply clones.

I hope, you already have your accounts on the Scratch website or RobboScratch3 installed. I hope, some of you have practiced at home to reinforce the skill of script making.

Please, open your games. Launch them. Make sure everything works.

Revision.

Now, please, tell me one by one what goes on in your games. If something doesn't work, show your screen, and we'll fix it together. The first one to notice the mistake should raise a hand and help their peer.

Let's revise basic concepts we have learnt at the previous lesson.

Slides No. 2-4

What is an algorithm? It's a sequence of simple actions (commands) of the object for successful achievement of the set goal.

What are the kinds of algorithms? When commands are carried out one after another: stand up, go, turn - it's a **Linear Algorithm**. If the algorithm repeats one and the same commands endlessly, it's a **Cyclic Algorithm**. If the algorithm carries out commands under some condition only, and doesn't do it otherwise, it's a **Branching Algorithm.**

What is a programming language? It's the language both humans and robots understand. This language is used to write programs for robots and sprites.

What is a program? It's an algorithm written in some programming language.

Look at slide 4. I will tell you elements of the interface. Please, write the figure this element is denoted with in the chat: blocks of commands (1), script (3), Start and Stop (2), Commands (4), Scene (5), Switch to Backgrounds (6), All the sprites are located here (7).

Let's continue developing our game.

Slide No. 5

Sensors. Timer. The game must have a goal. And you need to invent some challenges in the process of achieving this goal - otherwise it will not be interesting.

In our life, the most common challenge is a lack of time. It's the same in the game. Let's say, our goal is to gather all the sharks together. Let's add some thrill by limiting the time of the game!

Open the **Sensing** block. Find a beautiful ellipse called **timer**. The **Timer** shows how much time has passed since a certain moment. This moment is the start of the game. By switching the **timer** on, we can constantly see on the scene how much time has passed exactly - just set up the checkmark opposite it in the command section. The timer clock will appear on the scene immediately. It can be removed to a more convenient place.

The algorithm for game length settings can be as follows, for example: the game starts at 0 seconds, lasts 20 seconds and stops in 20 seconds.

To implement this algorithm we can use the **Conditional Operator**. Try to find it in the **Control** block of commands.

You encounter it in real life quite often. For example, your mother tells you that if you eat up your launch, you'll get sweets. Those who are told things like that, write 1 in the chat.

Formula for the timer: **If the Timer is more than 20, the game will stop.**

To apply the formula look in the **Operators** block of commands. There are green strips hexagon- and ellipse-shaped strips there. These are mathematical and logical operators (we'll discuss them a bit later). Let's find the **More** mathematical operator.

Put **timer** on the left in this operator and write 20 instead of 50 on the right. Then we'll put this small assembly into the hexangular **If …, then** cut out of the Conditional Operator.

Slide No. 6

These symbols are very much alike, even adults are often confused. That's why I decided to make a separate slide on them.

Where Is Greater, Where Is Less? *There's a very simple rule how to remember the difference between greater-than sign and less-than sign. Imagine there's a vertical line below. If the sign turns - in your head, mind! - into seven, it's a greater-than sign, and if it turns into four, it's a less-than sign.*

Slide No. 7

Put **timer** on the left in this operator and write 20 instead of 50 on the right. Then we'll put this small assembly into the hexangular **If …, then** cut out of the Conditional Operator.

We've got the branching operator that will go off 20 seconds after the start of the game. And to stop the game use the **Stop all** command from the **Control** block of commands. Add this command into the **If …, then** command.

The **reset timer** command must be put in the beginning of the script - this command allows launching the countdown in our game.

One timer is enough for the entire game if it's installed in the script of a sprite or even in the scene script. The timer's tendency to influence the entire game is called **Global**.

Add the condition of the game length to the forever Cyclic Operator, and... Whistle's gone, the game is on! In real games, a whistle is required for simultaneous beginning of the game for all players. We have assigned this role to the Check Mark. It is important to remember that when the shared check mark is clicked on, the game is launched and the timer is reset to zero at the same time.

Let's check how many seconds it will take for the game to stop. It's unclear if we have deleted more or less sharks during these 20 seconds. Besides, after disappearance for 3 seconds they reappear. It was funny to watch, but what we've accomplished is not obvious.

Eye training: follow the ball without turning your head (45 seconds)

https://scratch.mit.edu/projects/387341536

Slide No. 8

Variables. The result of a simple game is well visible due to the gamer's score. So, we need to teach our game to count points, for example, to award points, for example, one point for each caught shark.

It can be done with a **variable**.

Now we've come to the concept of a **Variable** - a critical one for all programmers.

Obviously, if there is a variable, there must be a constant somewhere. Both concepts refer to some numbers. Let's take age, for instance. E.g.: Age of 9. What is a variable and what is a constant here? The constant is 9. You'll be surprised, particularly those of you who were 9 last year, while now you are 10 or even going to turn 11 soon. Where is the constancy here? But the figure, the number itself never changes. If we change a figure, it will be another figure. While nine is always nine. We can always turn it upside down, for example, so that it becomes 6. But then it's 6 and no longer 9. As for the age, my age, it always remains mine and no one else's. But at same time, it's changing all the time, it's so VARIABLE! But it is changing with the help of CONSTANT numbers.

Making a Variable.

Making a variable will result in your computer saying: here's your variable! It will make some place for it in its memory. We don't know where this place is and we don't need to. The point is now this place has an address. Thought even this is not the main thing. The main thing is that we'll create this address ourselves. We'll create the **name for the variable** - that's what its address will be. We won't need to go there. That's the address for our program to go. Every time when it needs to change the value of the variable, the program will go to the new address, look how many scores, points, shots are there... It will have a look and add some more. Or throw away extras, on the contrary. You understand now that computer sees the variable as a box where it stores and changes our results.

Let's name our variable - Points. Now it has a name, so it's made.

Slide No. 9

Working with a Variable, part 1.

We need to decide which sprite will count points. If we assign this task to the Starfish, its sensor must convey the change of points at each touch with the Shark. I.e. the Conditional Operator should feature: **If touches** Shark, then **If touches** Shark1, then **If touches** Shark2, etc. We'll need to list all the possible touches of the Starfish.

And if we add a condition for the shark, one will be enough: **If touches** Starfish, then... Then we'll transfer this Conditional Operator with the change command to other sharks.

After we decide who is in charge of keeping count, we'll turn to making the Points variable.

A variable is made as follows. **Get in the Shark sprite.**

Get in the **Variables** block and click on **make a Variable**. Let's give the variable a name, even if no initial capital - it doesn't matter. It's set by default that this variable is valid for all sprites. Remember that in such a case the variable is **global**.

As soon as the variable is made it is displayed in the top right corner of the scene. There is a box next to its name with 0 inside. The computer has found a cell for the variable, and says it's empty. The main thing though is that now we have this cell. That's where the gained points will be accumulated.

Physical Activity Break. **Slide No. 22** Selecting Music.

Slide No. 10

The program counts clicks. Let's make a training program to reinforce the knowledge of a new concept, which is **'variable'**.

Add any sprite. It will be our training sprite. We'll delete it later.

Our game will also be a simple clicker. We will click on a sprite, and the Click variable will count how many times we have clicked. Think how we can do that. Correct. To start with, let's make the Click variable.

Then take a Check Mark in Events and put it under **Set Click 0**. We haven't made any clicks yet. Also, there is the title **When Sprite Is Pressed** in Events. That's what we'll need. What do we need to put under it? Correct. **Change Click for 1.** Let's check it. Click. Well, and if we want to take our clicks back, we can take another title - **When Space Bar Pressed** - and put **Change Click for -1** under it. Minus 1! Let's try. It works!

Slide No. 11

Working with a Variable, part 2. Let's go on with our underwater game. Out of 4 commands, two on the top are the most important for the variable. The first command sets the initial value of the variable, so the program uses this command to 'get to know' the variable. As the programmers say, it initializes the variable. Usually, this command is put into the starting conditions, i.e. between **the check mark** and the forever-cyclic operator. And since the variable is global, it's sufficient to perform initialization in one sprite only. It doesn't even matter which sprite it is. It can be background as well. However, while we are in the shark sprite, let's do it here.

Question to the chat. Do you think the initial value is always set as 0?

Right, you can start with different numbers, but points would better start with 0, of course.

Let's see how display of variables can be changed on the scene. Point on the table with the cursor, click with the RMB - right mouse button. The list drops down. Select the desired view with the LMB. The view with a tumbler allows setting the value of the variable by moving the tumbler. Want to award yourself with 20 points at once? Move the tumbler to the right, and you are a winner!

Slide No. 12

Working with a Variable 3. The second command on the top - **change … by**… - is the most important command for the variable. That's the command for counting our points. The counting algorithm is simple: when the starfish touches a shark, the shark disappears and the player gains +1. It works for each touch.

Unlike the **set...to…** command, the **change … by**… command must be included in each sprite involved in scoring. In case some shark has been missed, the score will change only due to the touch of the shark enabled by the **change … by**… command.

Check your code - have you put new commands correctly? Let's start. When the game stops (20 seconds after the start), the variable will show the results of our fishing.

You can compare your code with the code of the game - we've opened it in the beginning of the lesson. Click on the blue **Enter the Project** button, study the structure of each sprite and compare it with yours.

Slide No. 13

Sounds and Their Commands. You may say, there's no game without sounds, and I'll agree. We need to select the sound to accompany disappearance of the shark and another one - for its reappearance.

You already know where to look for sounds. Click on the **Sounds** button above the command field, and get in the **Sound Editor**. Get in the Sound Library and choose what you like. We have selected finger snap and pop sounds.

Commands from the **Sounds** block of commands are purple. Select **Start sound,** put it in the **if…then...** Conditional Operator. Put finger snap before the **wait** command, and the pop sound - after.

To liven up the game, let the events take place with gurgling sounds in the background. This sound is called **bubbles** in the Sound Library. We'll add this sound to the

Slide No. 14

Announcement of Results. Results of the game will be announced with the help of commands from the dark blue **Looks** block. We have used them while implementing the algorithm: **hide** and **show**.

Let the Starfish announce them. Get into its sprite.

First, place the **say hello! for 2 seconds** command in the script field. Replace 'hello' with any encouraging phrase. I suggest: 'Well done! Great fishing!' You can choose any other though. You can also increase the reading time at your discretion. Even half an hour if you like.

Question to the chat. How many seconds does half an hour contain? Yes! 1,800 seconds.

Now, after we've made our speech, it's high time to give your eyes some rest.

Eye Training (45 seconds).

Put the **say...** command into the **if...then...** Conditional Operator for the timer. By the way, there are only 5 of them instead of 20 on the slide. Why? Because we need to make sure everything works properly quickly. So programmers can reduce the number of trials for tests.

What monitors the result? Correct, the variable. So, let the variable announce the result. Let's add the **Points** variable to our phrase: Take the orange ellipse with 'points' written inside and put it into the **say hello! for 2 seconds** command. To check it just click on the assembly. The Starfish happily informs: 6. What is 6? Who gets 6, what is 6 for? Not clear. If you put the assembly with the variable in the script under the encouraging phrase, the Starfish will first say this phrase and then: 6.

Slide No. 15

String Operator. We want to see a simple and clear phrase: **Your score: 6.**

Get in **Operators** again. We already know mathematical operators. You can enter any numbers in their boxes. They don't get letters though. Try. You'll find string operators below mathematical and logical ones. We'll need the **join** command. As you can see, there are words in the boxes, not numbers. If you click on the command, it will print these words on the screen. Both words, one by one. At first, there were words 'apple' and 'banana'. Who can say what else is written there, aside from words? We can see that these two words are written separate, not solid. So, there's some invisible presence in one of boxes... What is it? Correct. It's a space. There is a space after 'apple'; otherwise, words would slur into one - unclear, though pretty tasty.

We want the results to be announced in the end of the game. **Your score: 6.**

The phrase will be the same for various results of the game, while the score will be different, it will be changed by our variable of Points. Don't forget the Space!

Slide No. 16

Announcement of Results 2. After combining the string and the variable put the **say** command into the assemble. Put all of it into the Conditional Operator under the command with an encouraging phrase that now can be erased. As you like.

Set the game length and the time for **Speak**.

The game is ready. Let's check it. Test it.

Slide No. 17

Clean-up of the Game. The game must be tested, so that you could fix or complete everything you don't like. I.e. the game needs to be in a marketable condition. It means something you've made looks nice enough for others to be used. You may like your game, while no one else want to play it. It means something is wrong with the game. Try to find out what. Have a look at other, popular games. Try to make your game no worse.

After 20 seconds the game must stop. In fact though, it's only the starfish that stops, while sharks go on swimming for a few seconds. That happens, because after 20 seconds the Starfish speaks for the whole 5 seconds. Sharks use this time for their last stroll. And they may touch the Starfish. The variable will grow, but the **say**... command has been already carried out, so the Starfish will 'say' the first result, while the points added while the Starfish has been talking will be changing in the table of variables. What a mess.

Let's fix it. Put **Stop all** in the **If the Timer is more than 20** Conditional Operator of each shark. But after you click on the white triangle, select **Stop This Script** instead of **Stop all** in the list. Then all the sharks will stop, and the Starfish will continue announcing your results, so that everyone could read it.

You can also arrange gradual coming of the tropical night in our sea.

In Scratch you can manage both properties of sprites and properties of background. We'll need the Appearance block of commands again. Note that there are less appearance commands for the background than for sprites. Identify what commands are lacking? Why? Any ideas?

Add the **Set effect brightness 0** command to the starting conditions. Don't see Brightness? Take **Set effect color 0**. You already know where to click. Select this very **Brightness** from a long list. **Brightness 0** means that the picture is drawn as it is. If you want to increase brightness, find a similar **Change effect brightness for 25** command. If you click on a sticker, i.e. increase brightness, the background will be brighter or, as one may put it, paler. If you replace 25 with minus 25 subtracting brightness, the background will get darker or grimmer or even turn black entirely.

Slide No. 18

Physical Activity Break. Arranged a few times during the lesson at the teacher's discretion.

The lesson is over! Thank you!

Doing Exercises in Workbooks, Answers

Exercise No. 1

5 < 6

Exercise No. 2

All. The name can be like that as well: without name. Though we see this ellipse, which means the variable has been made. And names can be different :-)

Exercise № 3

Pop sound.

Exercise No. 4

Change effect for brightness.