Lesson 1.4: The Race Track

Objectives

In this lesson, students will:

- ❖ Gain an understanding of global and local variables
- ❖ Practice implementing a timer
- ❖ Review the boolean concept and practice using boolean blocks
- ❖ Practice programming events, loops, conditional statements and creating variables

Agenda

1. The Race Track: Adding a Timer	15 mins
2. Moving Around the Track	10 mins
3. Student Activity: Moving Around the Track	10 mins
4. Challenge Activity: Kudos to the Winner	Time permitting
5. Wrap Up and Reflections	10 mins

Preparation

- ❏ Projector and speakers for class demonstration and instruction
- ❏ Computers connected to the internet
- ❏ Print student activity worksheet (one per student)

Resources & Links

- ❏ Solution Project: https://scratch.mit.edu/projects/545589461
- ❏ Starter Project: https://scratch.mit.edu/projects/544980195

© Code for fun
Released under the Creative Commons By Attribution license

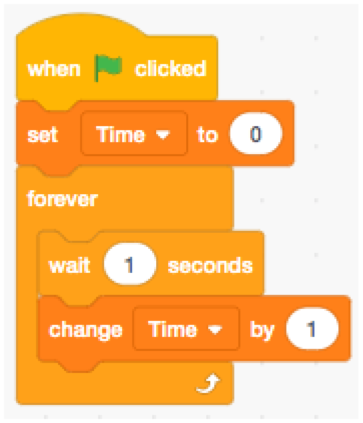
1. The Race Track: Adding a Timer

In this lesson students will code key events to move sprites around a race track. They will keep track of the time it takes each player to move the sprite to the finish line.

Display your screen and engage students in a demonstration and instruction of adding a timer to a project. Explain to students what they will be coding in this lesson.

Demonstrate the race track game using the solution project:
https://scratch.mit.edu/projects/545589461

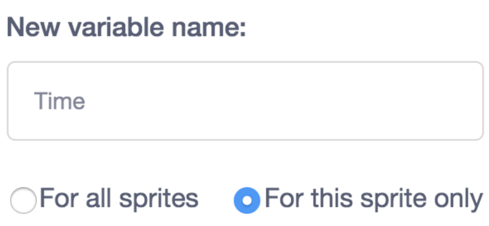
For each racer, we want to add the time it takes to get to the finish line. Scratch has a timer, but you cannot control it and it keeps going all the time. For our purposes, we will create our own timer. Which block in Scratch can help us keep track of time? Answer: The **Wait** block.



```
when [flag] clicked
set Time to 0
forever
    wait 1 seconds
    change Time by 1
```

If we want to keep track of seconds, we can create a timer variable, wait 1 second and then increase our timer variable by 1 . Let's take a look. (display the following code and explain it line by line).

Each racer (sprite) will track the time for their own timer. When we create the **Time** variable, we have 2 options:



New variable name: Time

For all sprites [✓ For this sprite only]

For all sprites is a global variable. Global variables can be read and changed by any sprite on the stage. **For this sprite only** is a local variable. Local variables can only be changed by their owner, but can be read by other sprites using the () of () block. When using local variables, you can use the same name because the variables are kept separate from each other.

Since each sprite needs to have its own timer, we are going to use a local variable. So we want to choose the "**For this sprite only**" option.

© Code for fun
Released under the Creative Commons By Attribution license

Keeping track of the time should be done in a separate script so it does not interfere with other code. When the program does multiple things at the same time, the program is doing things in parallel. In Computer Science we call it **parallelism** (say it a few times with students).

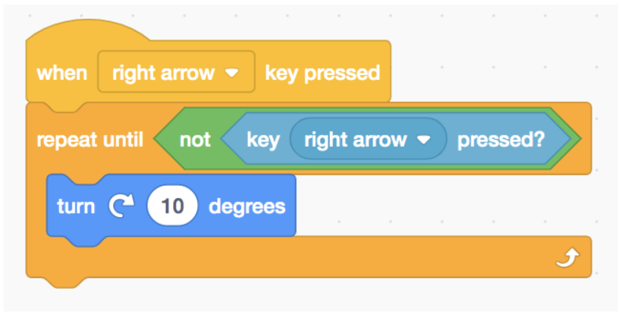
Student's Turn:

Instructions to give to students:

1. Remix the following project: 544980195
2. Click on the "**Player 1"** sprite and create a local variable (**For this sprite only**) called **Time.** Check the variable so it displays on the stage. Position the variable inside the middle of the track.
3. Add a **When green flag clicked** followed by the code to track time:
   a. Initialize the **Time** variable to zero
   b. In a forever loop, wait 1 second, then add 1 to the variable
4. Do the same for "**Player 2**". After you create the local variable called **Time**, copy the code from Player 1 (use the backpack).
5. Test your code each step of the way
6. The key stroke events for "Player 1" have been given to you. Try them out.

2. Moving Around the Track

Display your screen and engage students in a demonstration and instruction: Ask students what it was like moving player 1 around the track. The movement is not smooth and you can't turn and move at the same time. Also, the player can leave the race track. Let's fix these issues.



```
when [right arrow] key pressed
repeat until <not <key [right arrow] pressed?>>
    turn ↻ (10) degrees
```

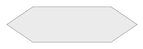
To smooth out the sprite's movements, we will allow the sprite to keep turning while a key is pressed. This is how we can do that:

In the sensing category is a block to check if a key is pressed. We want to continue moving until the key is **no** longer pressed, so we add the "**not"** operator. The "**repeat until"** block repeats something until a statement is true. Our

© Code for fun
Released under the Creative Commons By Attribution license

statement is: **not (right arrow key pressed)**. As soon as the player stops pressing the key, the statement becomes true and the loop stops repeating.

The “**key pressed”** and the “**not”** operator are boolean blocks. A boolean is a variable that can only have the value **true** or **false**. Boolean blocks in Scratch have the hexagonal shape:



Note: Explore other boolean blocks in Scratch to help cement the concept.

The next item we want to code is to keep the player inside the track. To do that, whenever the player touches the green area, the player must go back to the starting position. Since it is a beginner track, we will allow the **yellow** to be a buffer zone and not send the car back until the green area has been touched.

3. Student Activity: Moving Around the Track

Explain to students that in this activity they will be given pseudocode to guide them in writing the code for the activity. Pseudocode is a notation resembling a simplified programming language. It is often used by computer scientists to write down an algorithm without being tied to a particular programming language.

The following is an example of pseudocode:

```
When the sprite receives the “success” message
    Set the counter to 0
    Display a message
    Move the sprite to the bottom of the screen.
```

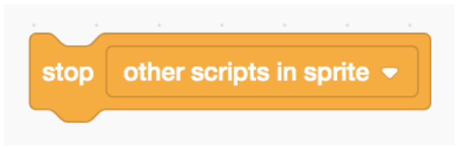
Explain the activity to students and distribute the student activity worksheet. Review the solution using the demonstration project.

4. Challenge Activity: Kudos to the Winner

It would make for a better race if the winner of the race would receive some kind of cheer.

The starter project has one backdrop if “Player 1” wins and one if “Player 2” wins. The challenge activity consists of changing the backdrop to show the winner when one of the 2 sprites crosses the finish line. The finish line is marked by a pink line. After the backdrop switches to announce the winner, the winner’s timer should stop. To do that we can use the following block to stop all other scripts, specifically the one tracking time:

© Code for fun
Released under the Creative Commons By Attribution license

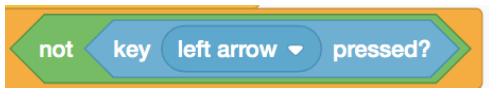


Explain the activity to students and instruct them to code it if they still have time after the previous activity.

3. Wrap Up and Reflections

Reflection Points:

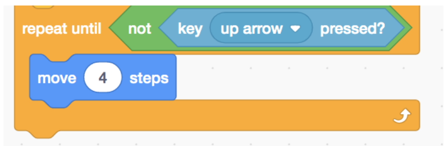
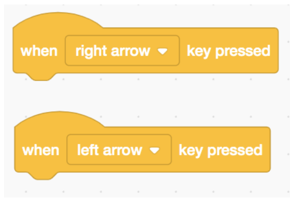
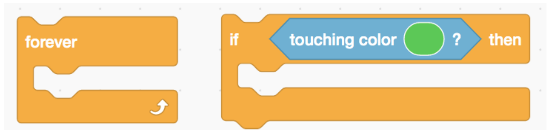
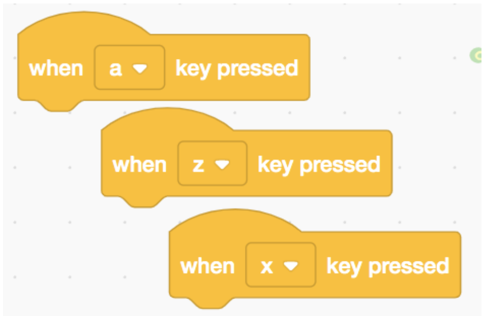
- What did you learn today?
- Which Scratch block is critical when keeping track of time in a variable? (Wait)



- How do you describe what this block does?
- What does the operator return when the left arrow key is pressed down? (false)

© Code for fun
Released under the Creative Commons By Attribution license

Student Activity: Moving Around the Track

What to do:	Using/Details:
In your race track project, for "Player 1" change the up arrow key event script so that • **Repeat until** the up arrow key is **not** pressed. • Move 4 steps	 repeat until not key up arrow pressed? move 4 steps
Change the key event code for the **right** and **left arrows** as well	 when right arrow key pressed when left arrow key pressed
After the initialization code add: In a forever loop: If touching green Go back to the starting position	 forever if touching color ? then
Copy the code from "Player 1" to "Player 2". Use different keys for "Player 2". Keys on the other side of the keyboard are a good choice so both players can play. The starting position is also a little different.	 when a key pressed when z key pressed when x key pressed
Test your code after every change.	

© Code for fun
Released under the Creative Commons By Attribution license