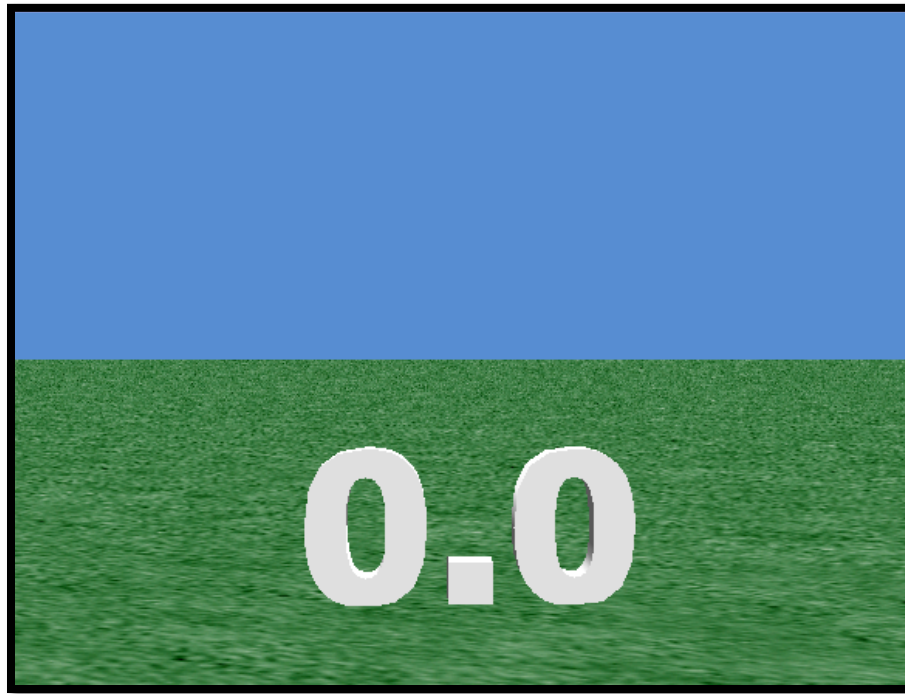


Making a Timer in Alice



By Jenna Hayes
under the direction of Professor Susan Rodger
Duke University
July 2008

www.cs.duke.edu/csed/alice/aliceInSchools

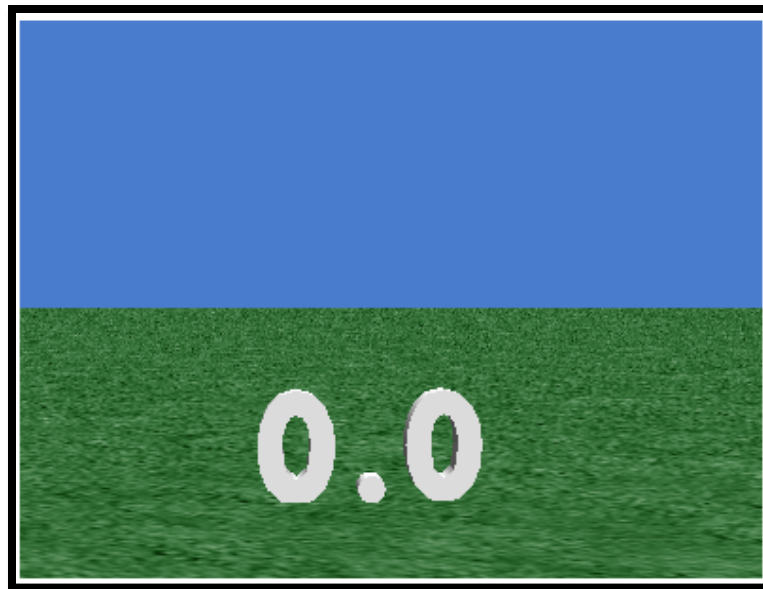
Step 1: Adding the Text Object

This tutorial will teach you how to make a timer in Alice. Timers can be very useful if you are interested in making timed games and have many other uses.

Start a new Alice world, and add a **text object** to that world. When it asks you what you want the text object to say, type in **0.0**.



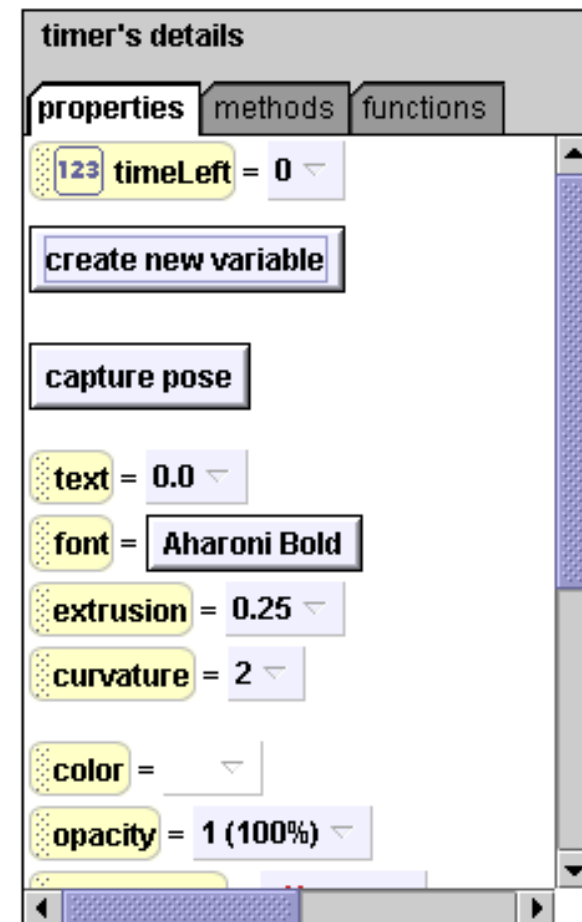
Now in your object tree right click on **0.0** and rename it **timer**. Let's get started coding our timer.



Step 2: Creating a Variable

Click on **timer** in your object tree, and then go to the **properties** tab. Click on the **create new variable** button. Create a **Number** variable named **timeLeft**. For now, set its value to **0**.

That 0 is just a placeholder. We will write code in the method editor so that we can enter in whatever value we want before we play the world.



Step 3: Initialize Method

Create a class-level method for **timer** called **initialize**. The only command we'll need in this method is one that sets the value of **timeLeft**. So click on **timeLeft** and drag it into the **initialize** method. Set its value to **1** for now.

The screenshot shows a programming environment with two tabs: 'world.my first method' and 'timer.initialize'. The 'timer.initialize' tab is active and shows a method definition with a parameter 'amountOfTime' (value 123). Below the parameter is the text 'No variables'. A block is placed in the method body: 'timer.timeLeft' (with a dropdown arrow) followed by 'set value to' and '1' (with a dropdown arrow), and 'more...' (with a dropdown arrow). On the right side of the editor, there are two buttons: 'create new parameter' and 'create new variable'. At the bottom of the editor is a palette of control blocks: 'Do in order', 'Do together', 'If/Else', 'Loop', 'While', 'For all in order', 'For all together', 'Wait', 'print', and a comment block '//'.

Step 3: Initialize Method

Now create a **number parameter** in **initialize** called **amountOfTime**. Drag and drop it over the **1** in your **set value to** command. Now we can set the number to a different value every time we use a timer, without having to change the **initialize** code.



Now drag your **initialize** method into **world.my first method** so that it happens right when your world starts. Set **amountOfTime** to any number you want.

Step 4: Count Down Method

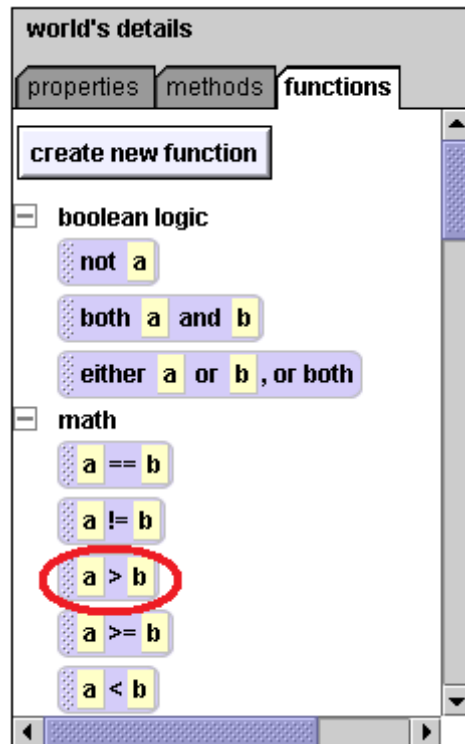
Now we need to write a method that will decrement the `timeLeft` variable, and have our text object display the seconds as they tick down. Create another class-level method, called `countDown`. Drag a **Do in order** inside the method, and then drag a **While** loop inside that.

The screenshot shows a programming environment with three tabs: `world.my first method`, `timer.initialize`, and `timer.countDown`. The `timer.countDown` tab is selected and active. The editor area for this method shows:

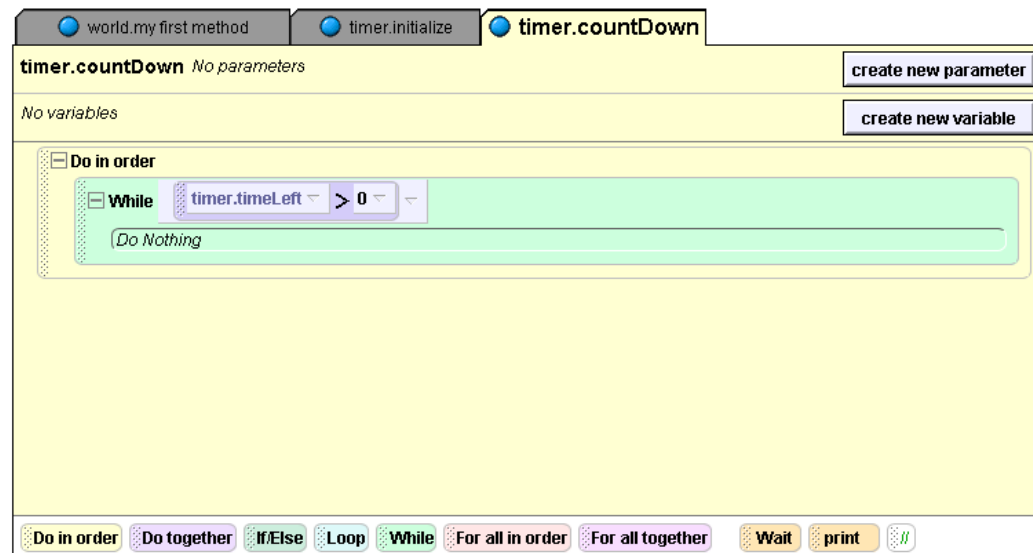
- Method name: `timer.countDown` *No parameters*
- Variables: *No variables*
- Control blocks:
 - Do in order** (containing)
 - While true** (containing)
 - Do Nothing*

At the bottom, there is a palette of control blocks: `Do in order`, `Do together`, `If/Else`, `Loop`, `While`, `For all in order`, `For all together`, `Wait`, `print`, and a comment block.

Step 4: Count Down Method



Click on **world** in your object tree and then click on the **functions** tab. Find the **a > b** button under **math**.

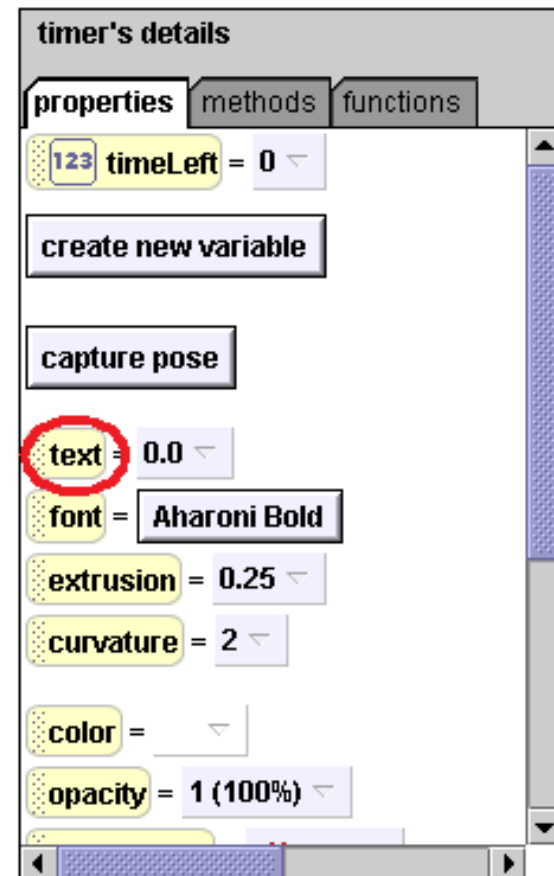


Drag that button over the **true** part of your **While** loop; choose any values, we are going to replace them. Now find **timeLeft** in the **timer's properties** tab, and drag it over **a**.

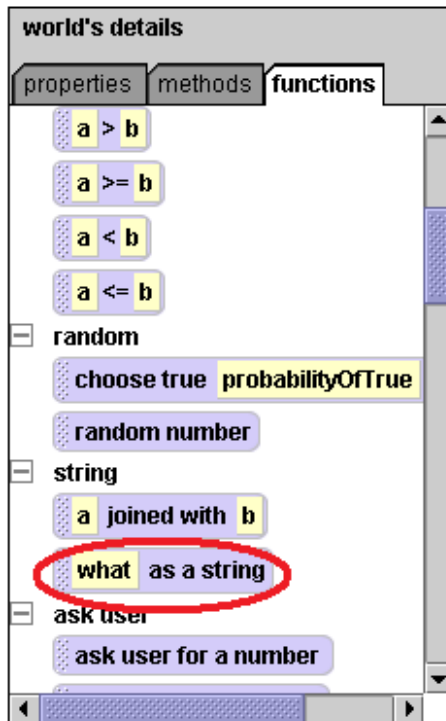
Step 4: Count Down Method

Drop a **Do in order** inside the **While** loop. Now we need to change the text of our text object every time **timeLeft** changes. Click on **timer** in the object tree and then click on the **properties** tab. You should see the **text** button.

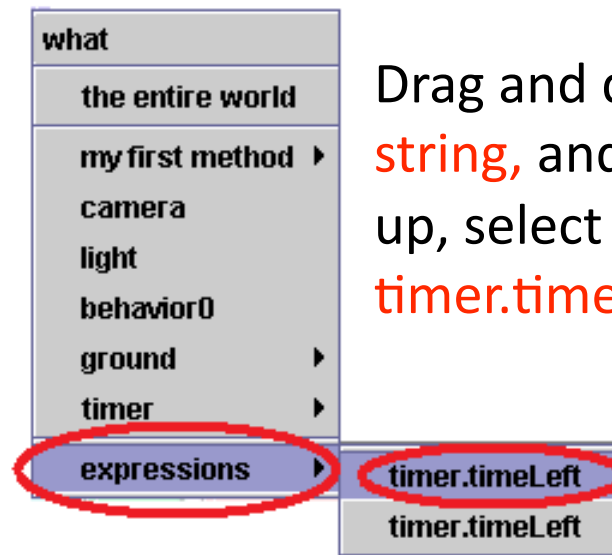
Click that button and drag it into the **Do in order** inside the **While** loop. Set it to **default string** for now.



Step 4: Count Down Method

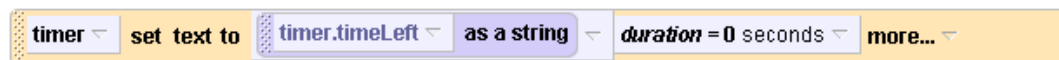


Now we need to turn `timeLeft` into a `string`, so we can display it with our text object. To do this, click on `world` and then the `functions` tab, and scroll down until you see `what as a string`.



Drag and drop that over `default string`, and when the menu pops up, select `expressions` and `timer.timeLeft`.

Set the duration of this command to `0 seconds` so that it's value is set instantaneously.



Step 4: Count Down Method

Now we need to make sure that it takes exactly one second before the value of `timeLeft` is reset. Drag the **Wait** command, which is located under your method editor, into your **Do in order** in your **While** loop and set it to **1 second**.

The image shows a Scratch code editor window with two tabs: 'world.my first method' and 'timer.countDown'. The 'timer.countDown' tab is active, showing a method editor with the following structure:

- Method name: `timer.countDown` (No parameters)
- Variables: No variables
- Control: **Do in order** (expanded)
 - Condition: **While** `timer.timeLeft > 0` (expanded)
 - Control: **Do in order** (expanded)
 - Block: `timer` **set text to** `timer.timeLeft` **as a string** `duration = 0 seconds` **more...**
 - Block: **Wait** `1 second`

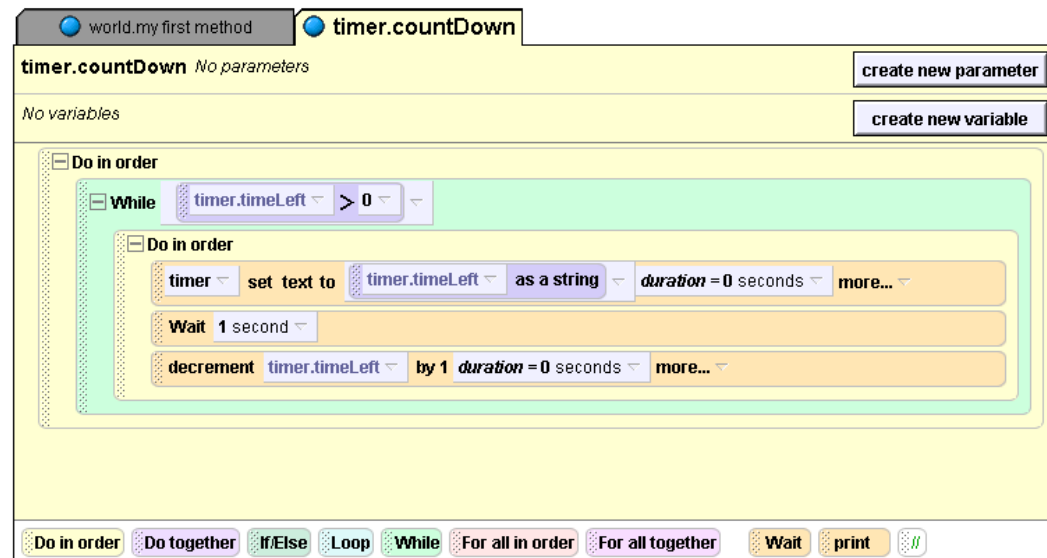
A red arrow points from the **Wait** block in the code editor to the **Wait** block in the bottom toolbar, which is circled in red. The bottom toolbar also includes **Do in order**, **Do together**, **If/Else**, **Loop**, **While**, **For all in order**, **For all together**, **print**, and a comment icon.

Step 4: Count Down Method

Click on **timer** in the object tree, and then go to the **properties** tab. Click on **timeLeft** and drag it into your method editor right under your **Wait** command. On the menu that pops up, chose **decrement timer.timeLeft by 1**.

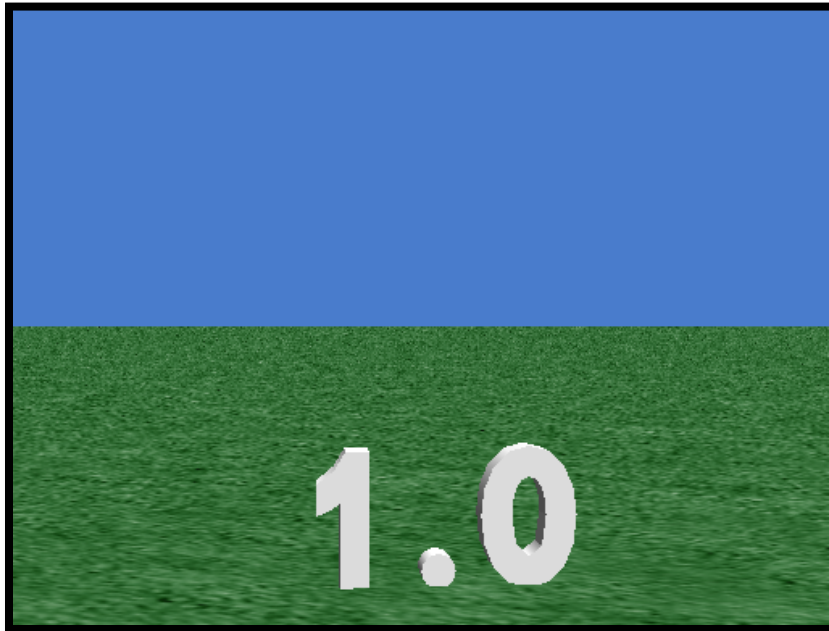


Now, so that the timer is decremented instantaneously, set the duration of the decrement command to **0 seconds**.



Now drag your **countDown** method into **world.my first method** under your **initialize** method and **play** your world to see what happens.

Step 5: Finishing Up



Notice anything strange about your timer? No matter how many seconds it starts with, it always stops at 1!

This is why: Look at your **While** statement. It will only repeat itself if **timeLeft** is greater than zero at the beginning of the statement. When **timeLeft** gets down to zero, the **While** statement stops and the text object is never reset. So we need to add a command AFTER your **While** statement so the timer goes all the way down to 0.

Step 5: Finishing Up

Go to **timer** on the object tree and then go to the **properties** tab. Find the **text** button and drag it into your **countDown** method under your **While** statement. Reproduce the same **timer set text to** command that you have inside your **While** statement. Your final code will look like this:

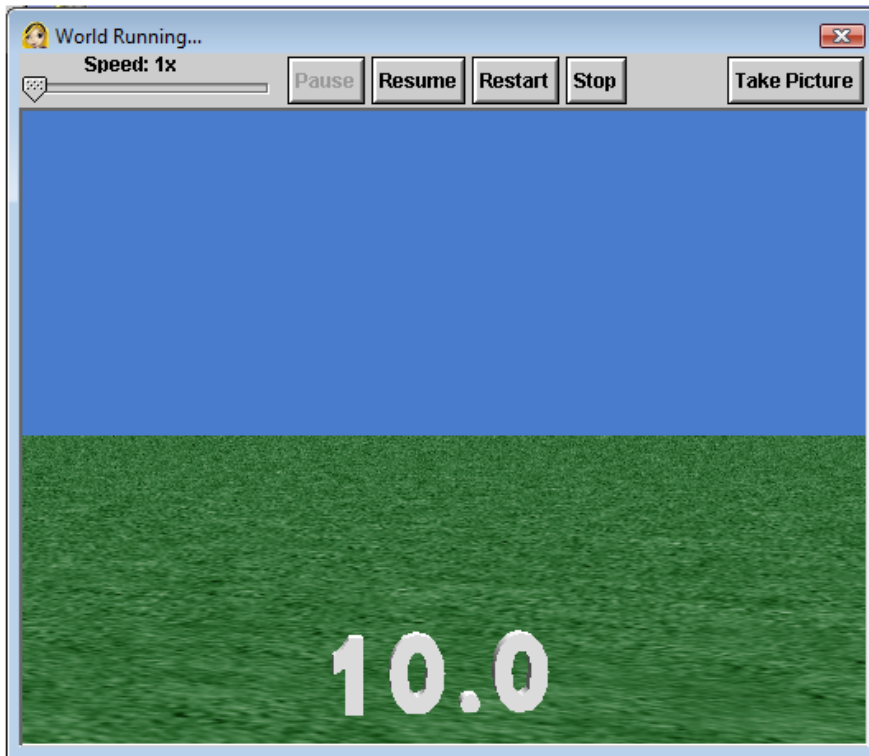
The screenshot shows the Scratch code editor for the **timer.countDown** method. The code is structured as follows:

- Do in order** block:
 - While** loop with condition `timer.timeLeft > 0`:
 - Do in order** block:
 - timer** **set text to** `timer.timeLeft` **as a string** `duration = 0 seconds` **more...**
 - Wait** `1 second`
 - decrement** `timer.timeLeft` **by** `1` `duration = 0 seconds` **more...**
 - timer** **set text to** `timer.timeLeft` **as a string** `duration = 0 seconds` **more...**

The bottom of the editor shows a palette with various control blocks: **Do in order**, **Do together**, **If/Else**, **Loop**, **While**, **For all in order**, **For all together**, **Wait**, **print**, and a **Run** button.

Now **play** your world again, and observe that sweet sweet timer action!

Step 5: Finishing Up



This timer can be very useful for games in which you have to beat the clock. Your timer will need to be run in a **Do Together** with the other code for your game, or as a separate event in your game.

You can also use these concepts to create a scorekeeper (see my scorekeeper tutorial for more information).