Challenge Lesson: Ozobot Bit— The Coin Change Maker

Created by

Richard Born

Associate Professor Emeritus

Northern Illinois University

richb@rborn.org

Topics

Coin change

Algorithms

Computer Science

Functions

Ages

Grades 9-12

Duration

Depending upon familiarity with Mode 4 OzoBlocks, this could take a few hours. Best assigned as a take home project to be due a number of days after assigning it, so students could work on it at their "leisure."

Challenge Lesson:
Ozobot Bit—The Coin Change Maker

By Richard Born
Associate Professor Emeritus
Northern Illinois University
rborn@niu.edu

Introduction

Before the advent of computers, making proper change for a sale at a store was the job of the clerk. If the bill for the purchased items was, for example $1.59, and the customer gave the clerk $2, the clerk would have to figure out (1) how much the change should be, and (2) what denominations of coins to give the customer in return. The clerk would mentally subtract 59 from 100 to determine that the amount of change should be 41 cents. There are, of course, many ways to make the change: four dimes and a penny; or a quarter, dime, and six pennies; or even 41 pennies, which would probably make the customer somewhat unhappy. In one sense the best way to make the change would be to give the customer a quarter, dime, nickel, and penny.

This last way is the generally accepted way to make coin change, assuming that there is a sufficient supply of each of the four coin denominations. Note that we are also assuming that the half-dollar coin does not enter the picture, as its circulation is quite small in this day and age. (If the change called for two quarters, and the clerk happened to have a half-dollar, it could be used as an alternative to two quarters.) This algorithm for making change looks at the coins in order from largest to smallest denomination, first determining the number of quarters, then the number of dimes in the remaining change, then the number of nickels in what remains, and finally the number of pennies.

We've all been at a grocery store, fast-food restaurant, or supercenter store, to mention just a few, and grabbed our change from a dispenser attached to a point-of-sale terminal. There is a good chance that the point-of-sale terminal used the above algorithm to determine your coin change. So the question in this day and age is "how is this coin change algorithm programmed." Designing the program for this algorithm is an excellent exercise for any student of computer science. What better way to do this than to program Ozobot Bit to determine the change amount and then display the number of quarters, dimes, nickels, and pennies making up the change!

There are two OzoBlockly blocks that are particularly helpful when implementing the coin change making algorithm. They are shown in Figure 1.

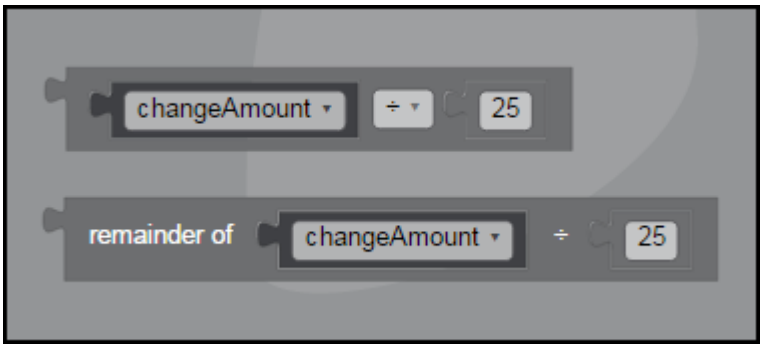


Figure 1

You need to remember that OzoBlockly division is integer division, and as such the division process results in the quotient with the remainder missing. Thus if the *changeAmount* was 68, the result of the division would give a quotient of 2, the number of quarters required in the coin change! The remainder when *changeAmount* is divided by 25 would be 18. 18 could then, in turn, be divided by 10 to determine the number of dimes, and so on through nickels and pennies.

The Map for this Challenge Lesson

The real challenge when designing your OzoBlockly program is to come up with a way in which Ozobot Bit can display the billed cents and the proper coin change after subtracting the billed cents from 100. In this program, we are only concerned with the coin change that would be dropped into the coin dispenser, not with any paper money change returned to the customer by the clerk. Figure 2 shows a small copy of the map that we will use to discuss how your program should handle Ozobot Bit's behavior. A larger copy of the map, suitable for printing and using with your Ozobot Bit and your OzoBlockly program, can be found on the last page of this document.

You will immediately notice that the map consists of two main sections. In the black portion at the top, Ozobot Bit will display an aqua LED while moving about. *XY* is a random two digit number between 0 and 99, representing the billed cents. Ozobot Bit will stop at the *X* location and blink a red LED the number of times equal to the value of *X*. He will then stop at the *Y* location and again blink a red LED the number of times equal to the value of *Y*. As an example, the value of *$.XY* might be $.32. This is the billed cents, from which Ozobot Bit would calculate the coin change due the customer as 100 - 32 = 68 cents.

Ozobot bit will then move to the black line at the intersection on the far left, and make a left turn. There he will delay for 15 seconds, giving the *user* a change to determine (on a piece of scratch paper, if necessary) the proper way in which to make the coin change. Ozobot bit will then move to the green portion of the map by stopping at a number between 0 and 4 and

displaying a greed LED for one second, thus displaying the number of quarters, dimes, nickels, and pennies required for the proper change.

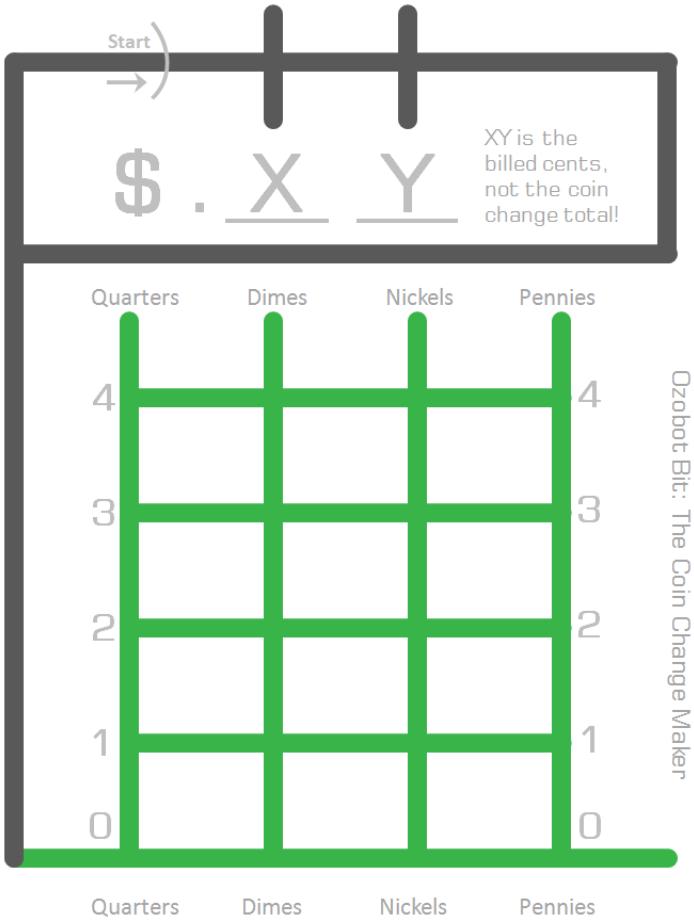


Figure 2

Some Suggestions

This challenge is complex enough that the use of functions (also known as procedures) will help to break the code into more manageable parts. In addition, it will keep the number of OzoBlocks required and Ozobot memory for storing programs smaller.

The main program could perform tasks including:

- Setting the initial LED color to aqua, and line-following speed to 60 mm/sec.
- Get a random integer between 0 and 99 as the *billedCents*.
- Use integer division and the "remainder of" OzoBlock to compute the *X* and *Y* values indicated at the top of the map.

- Stop Ozobot at the *X* and *Y* locations and have him display the values of *X* and *Y* by blinking the LED a red color the appropriate number of times.
- Have Ozobot travel to the black intersection of the far left of the map, turn left, and wait for 15 seconds. As stated earlier, this is to give the user some time to calculate the correct way to make change, so it can be compared to Ozobot's algorithm for making change.
- Define a function that you might name *makeChange* and call that function. See Figure 3 below.

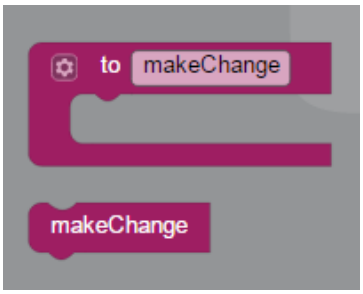


Figure 3

The code for the "*to makeChange*" function would use the algorithm discussed earlier for determining the number of *quarters*, *dimes*, *nickels*, and *pennies* in the coin change. Another function, which you might call *displayChange,* would then be used to navigate Ozobot bit on the bottom green portion of the map. See Figure 4 for the function definition and how it might be called.

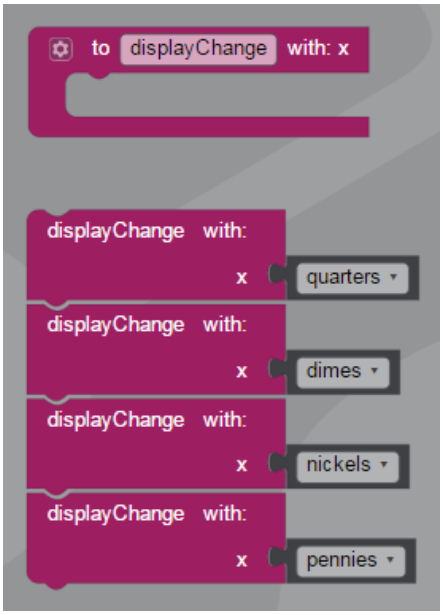


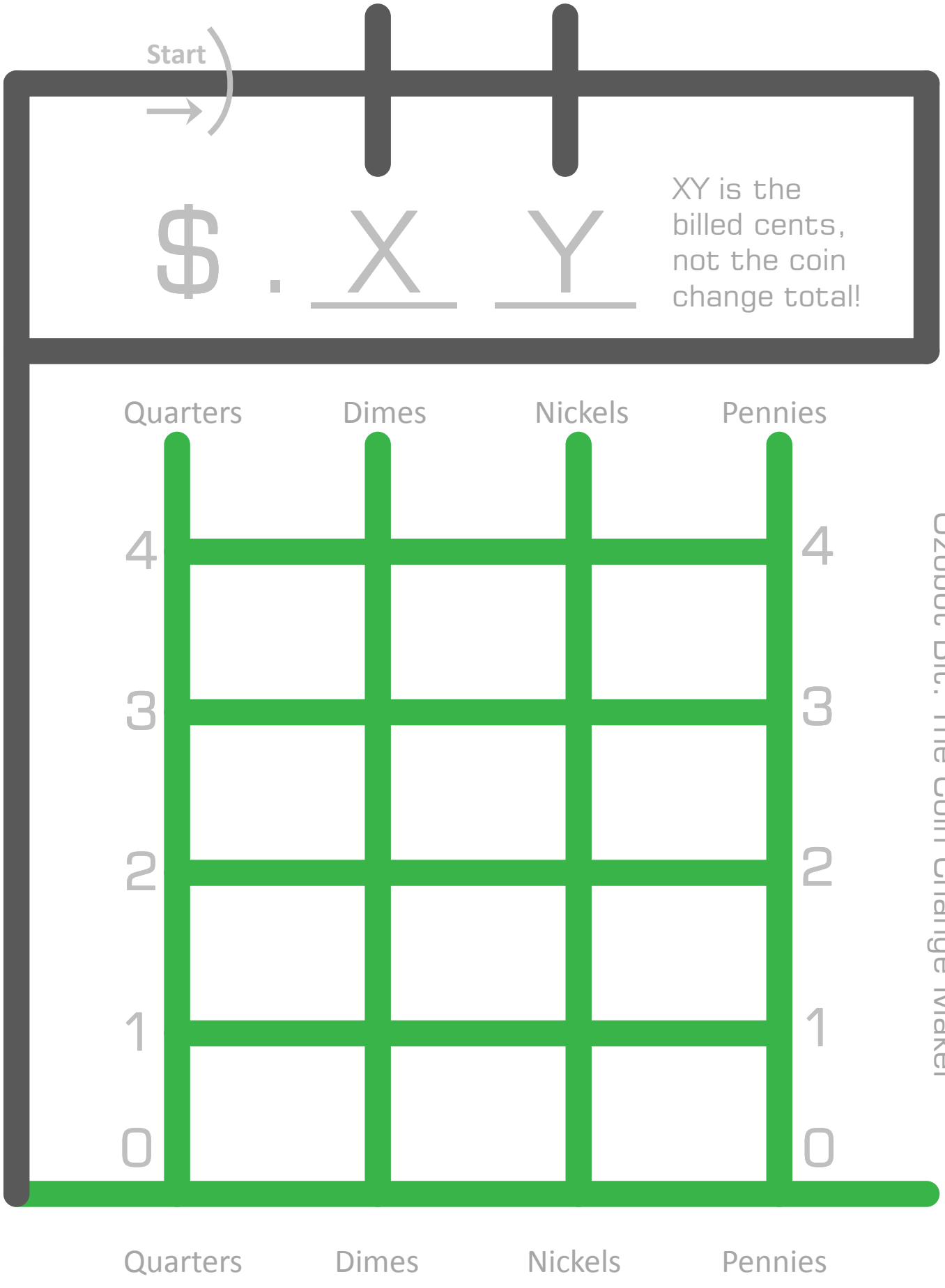
Figure 4

By using the "*displayChange with x*" option, you can *avoid repeating the code four times* when navigating Ozobot as he displays the values for the number of quarters, dimes, nickels, and pennies in the coin change. You will need to make code in such a way that Ozobot Bit will travel up to the appropriate green horizontal line, show a green light for 1 second, then turn around and go down the vertical green line, and proceed to the next green denomination's green vertical line.

At this writing, Level 5 (Master) is not yet available. It is supposed to contain some blocks related to the use of arrays. If you program this challenge lesson using arrays or other Level 5 features, then your program may be significantly different than outline in the above suggestions.

Challenge Lesson Requirements

1. Use OzoBlockly codes from Mode 4 (Advanced) or higher.
2. Make sure that you calibrate Ozobot Bit on paper before testing and running your program.
3. Make sure that Ozobot Bit's battery is fully charged and the wheels are clean.
4. To start Ozobot Bit's program, you need to double-press the start button.
5. Ozobot Bit should be started where you see "Start" in the upper area of the maze, facing the direction shown by the gray arrow.
6. Ozobot Bit should behave in the manner described on the previous pages of this document.



Ozobot Bit: The Coin Change Maker