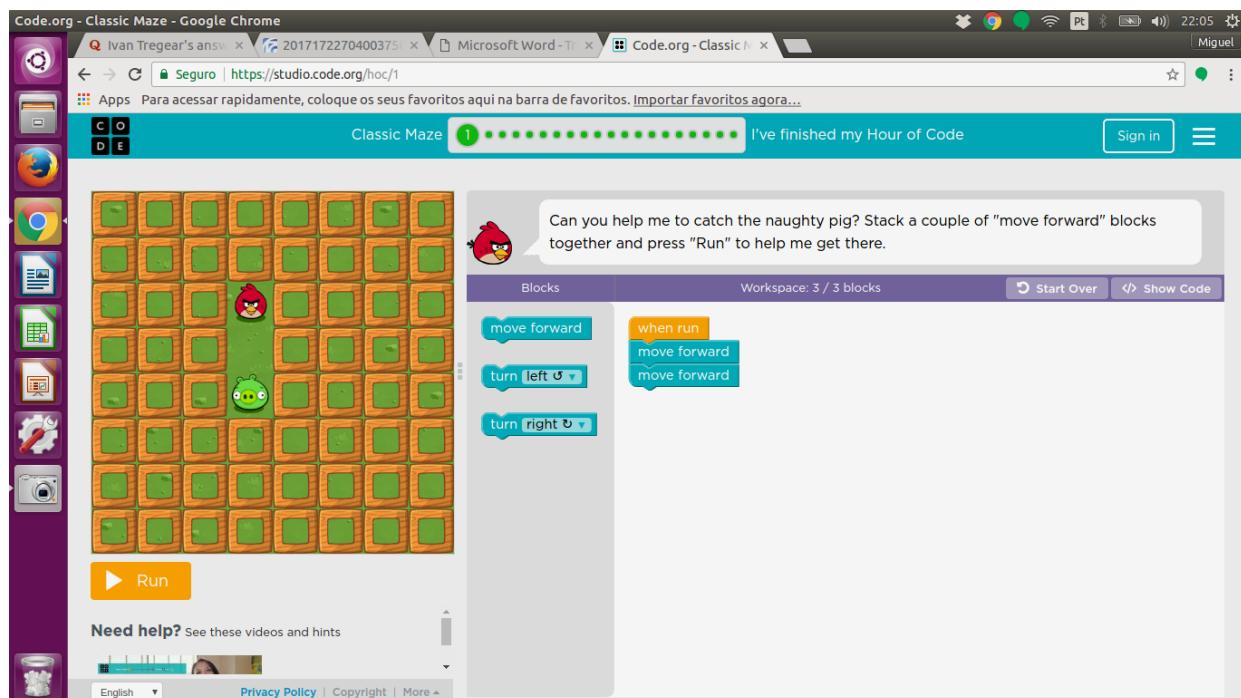


UFMT- UNIVERSIDADE FEDERAL DE MATO GROSSO  
CIÊNCIA DA COMPUTAÇÃO  
DISCIPLINA: PROGRAMAÇÃO I  
DOCENTE: IVAIRTON M. SANTOS  
DISCENTE: MIGUEL FREITAS DE LIMA TURMA: CC 2017/1  
DATA:28/06/2017

## TRABALHO 1

### DESAFIO 1.



Analisado o desafio, observei que o pássaro precisaria fazer apenas dois movimentos retos para chegar até o porco, então inseri no workspace dois comandos “move forward”.

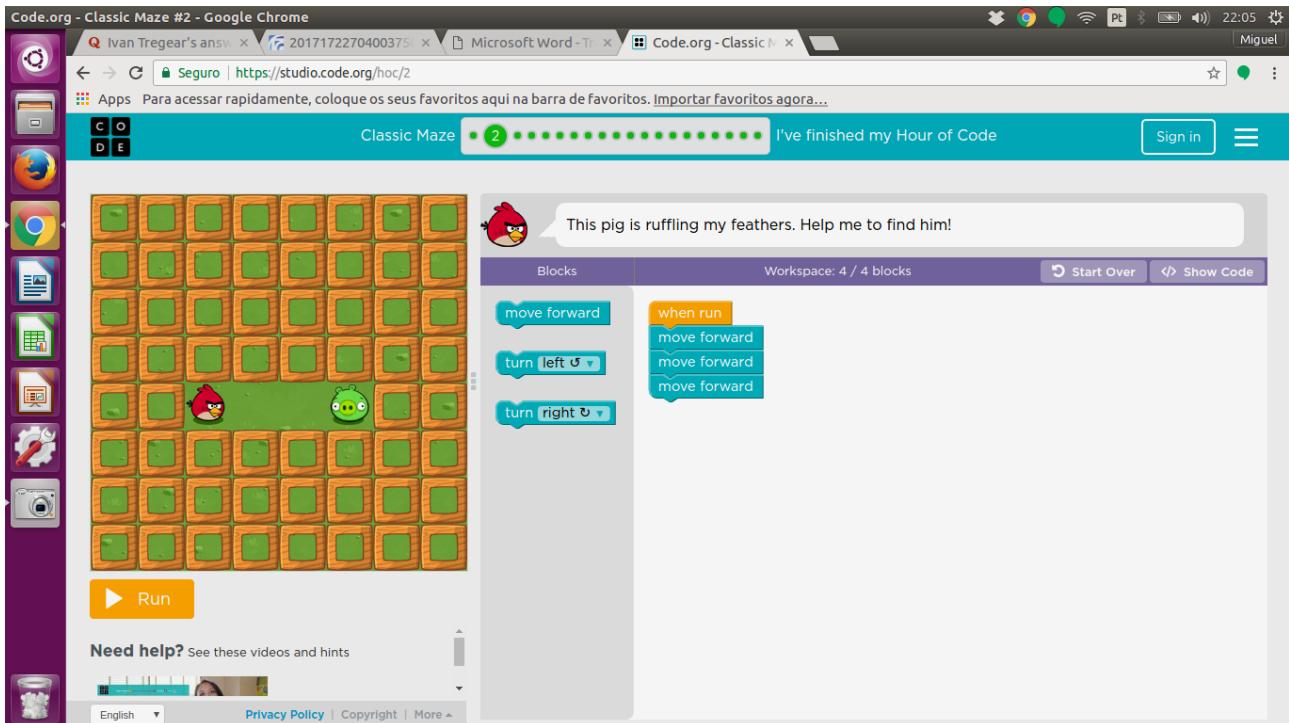
Código fonte em Java Script:

```
moveForward();
```

```
moveForward();
```

O algoritmo é simples, e foi utilizado apenas dois comandos.

## DESAFIO 2.



Analisando o desafio, observei que o pássaro precisaria fazer apenas movimentos retos, descartando os comandos de virar, porém diferente do primeiro desafio, esse exigiu um comando “move forward” a mais.

Código Fonte em Java Script:

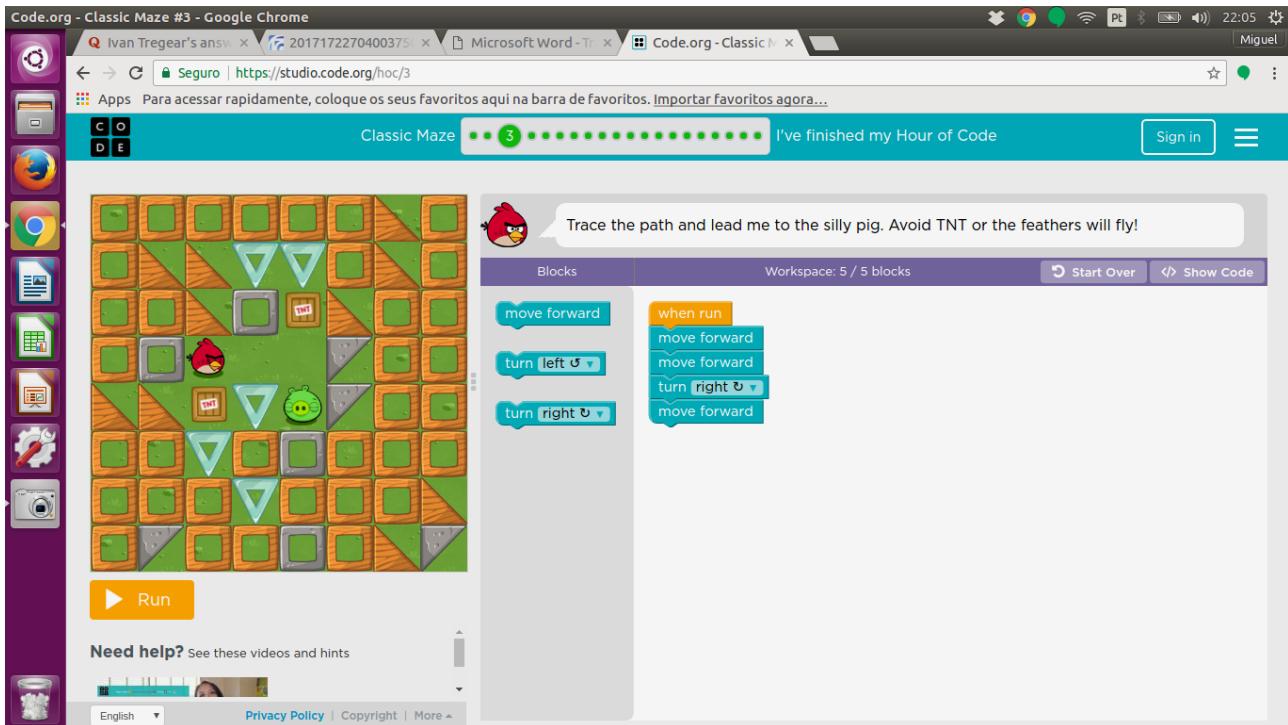
```
moveForward();
```

```
moveForward();
```

```
moveForward();
```

O algoritmo exigiu um comando a mais em relação ao do primeiro desafio.

### DESAFIO 3.



Analisando o desafio, observei que agora haveria a necessidade do pássaro virar a direita após mover-se duas vezes para frente, e depois mover-se novamente para a frente. Então usei um comando que ainda não havia sido usado “turn right”. Organizei os blocos de comando de forma adequada para o pássaro chegar ao porco.

Código fonte em Java Script:

```
moveForward();
```

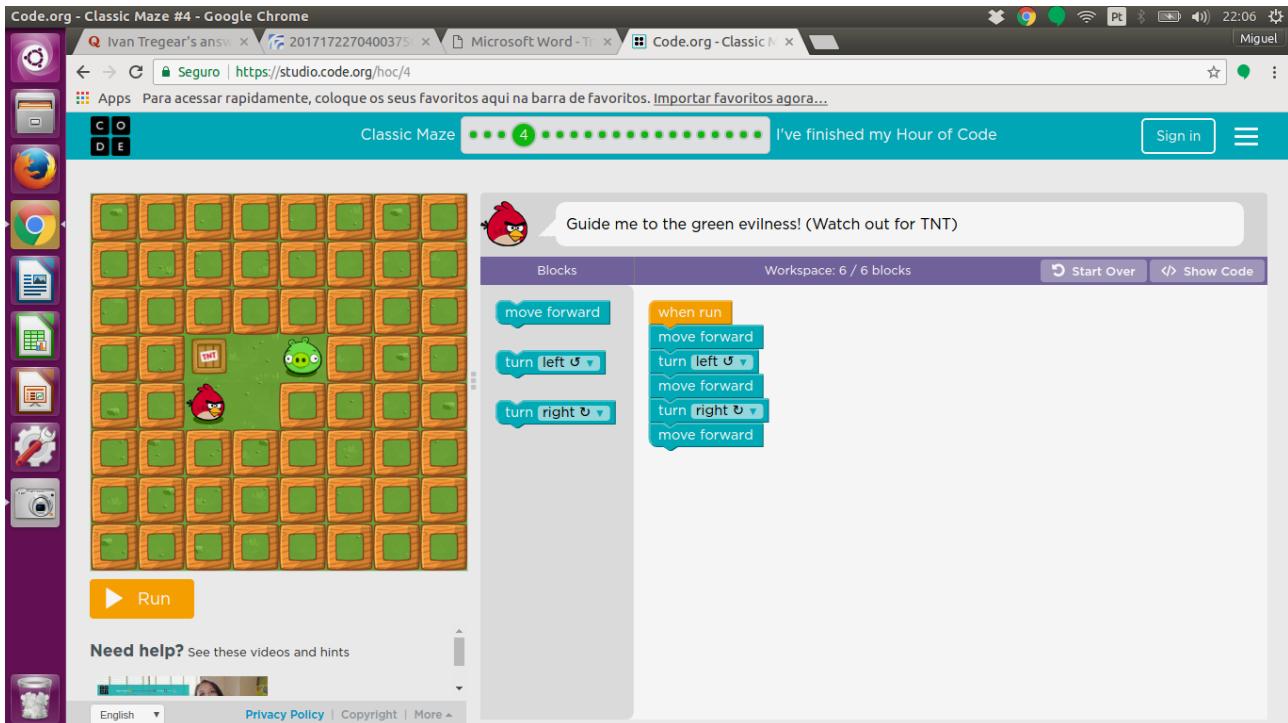
```
moveForward();
```

```
turnRight();
```

```
moveForward();
```

O algoritmo continua usando uma sequência de comandos simples para chegar ao porco, agora usando o novo comando “turnRight();”

## DESAFIO 4.



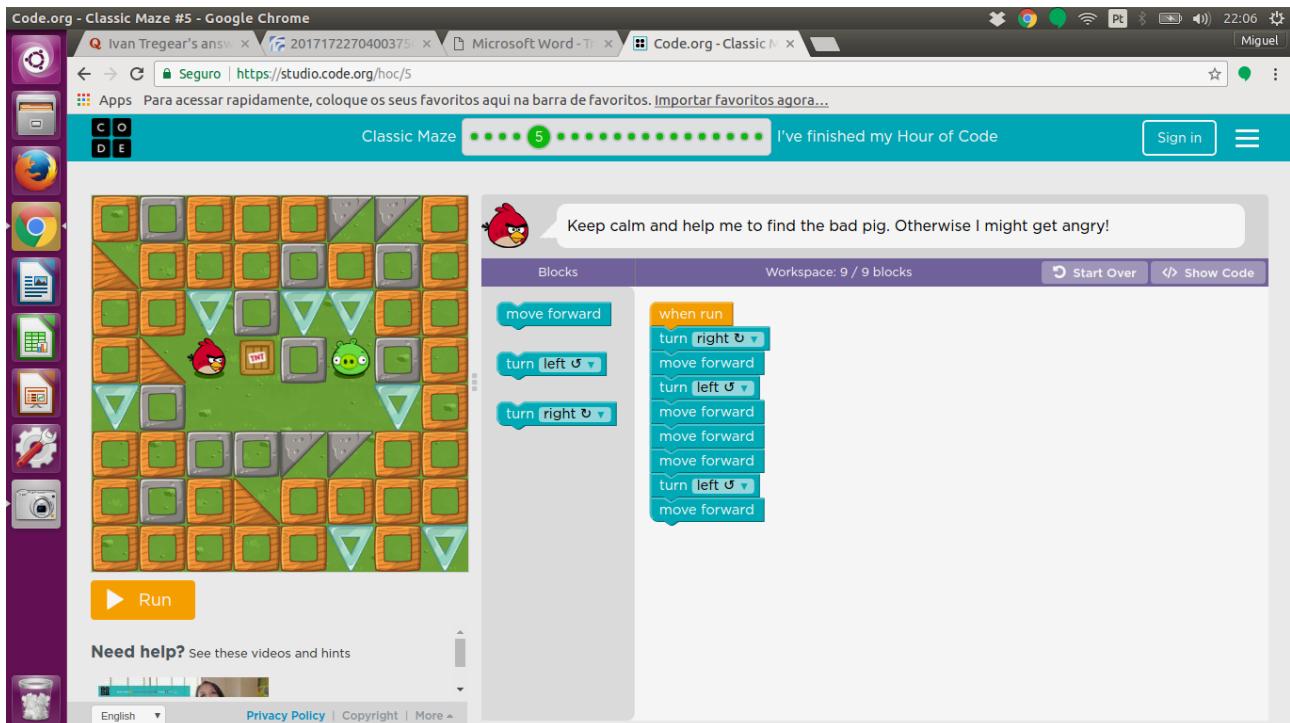
Analisando o desafio, observei a necessidade que fazer duas viradas para lados diferentes, então usei o comando “turn” tanto para à direita quanto para à esquerda.

Código fonte em Java Script:

```
moveForward();  
  
turnLeft();  
  
moveForward();  
  
turnRight();  
  
moveForward();
```

O algoritmo coloca os comandos em ordem, de forma simples e bastante compreensível.

## DESAFIO 5.



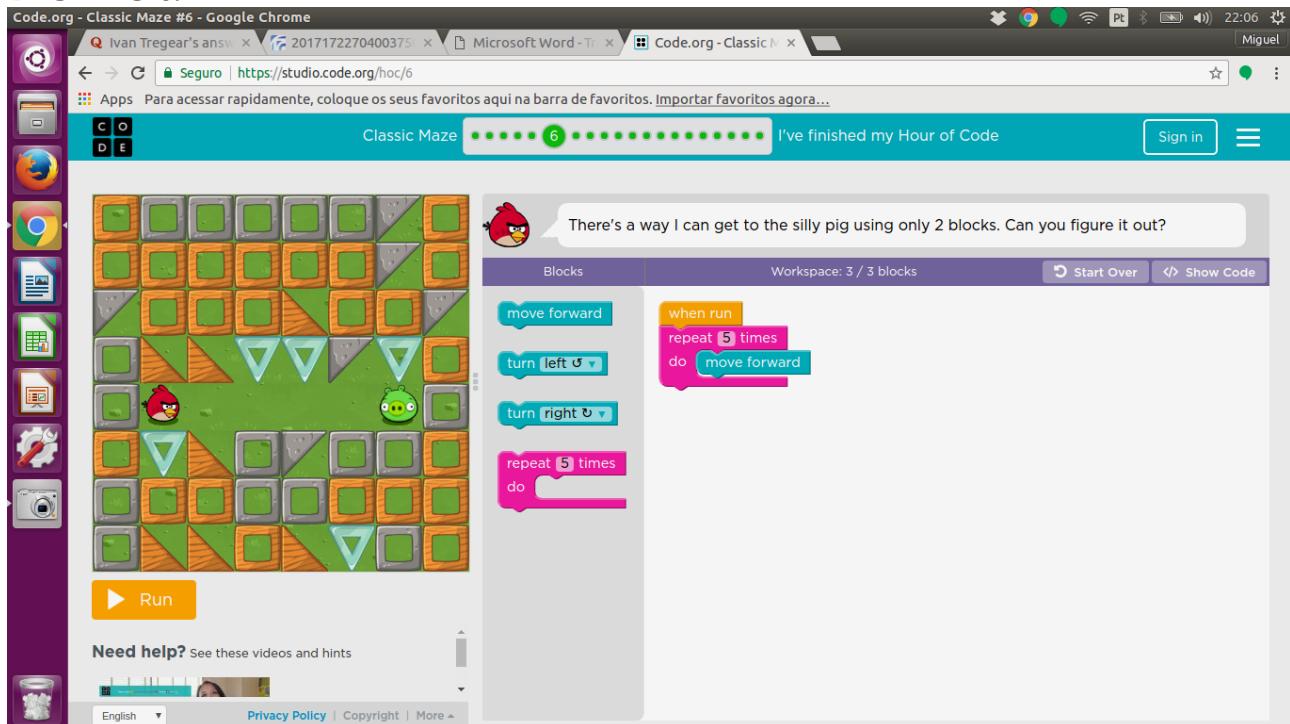
O bloco de comandos é organizado igual aos outros, com a repetição dos comandos já usados anteriormente. Os blocos são alocados de forma que o pássaro vá até o porco sem colidir em nada.

Código fonte em Java Script:

```
turnRight();  
  
moveForward();  
  
turnLeft();  
  
moveForward();  
  
moveForward();  
  
moveForward();  
  
turnLeft();  
  
moveForward();
```

O algoritmo usa os comandos já expostos nos desafios anteriores, porém com repetição de acordo com as necessidades do desafio.

## DESAFIO 6.



Nesse desafio foi exposto um novo tipo bloco de comando o “repeat” conhecido como laço. No lugar de repetir 5 “moveForward” foi possível utilizar o laço determinando que deveria ser repetido 5 vezes os comandos ali dentro. Diminuindo assim o uso de comandos repetitivos.

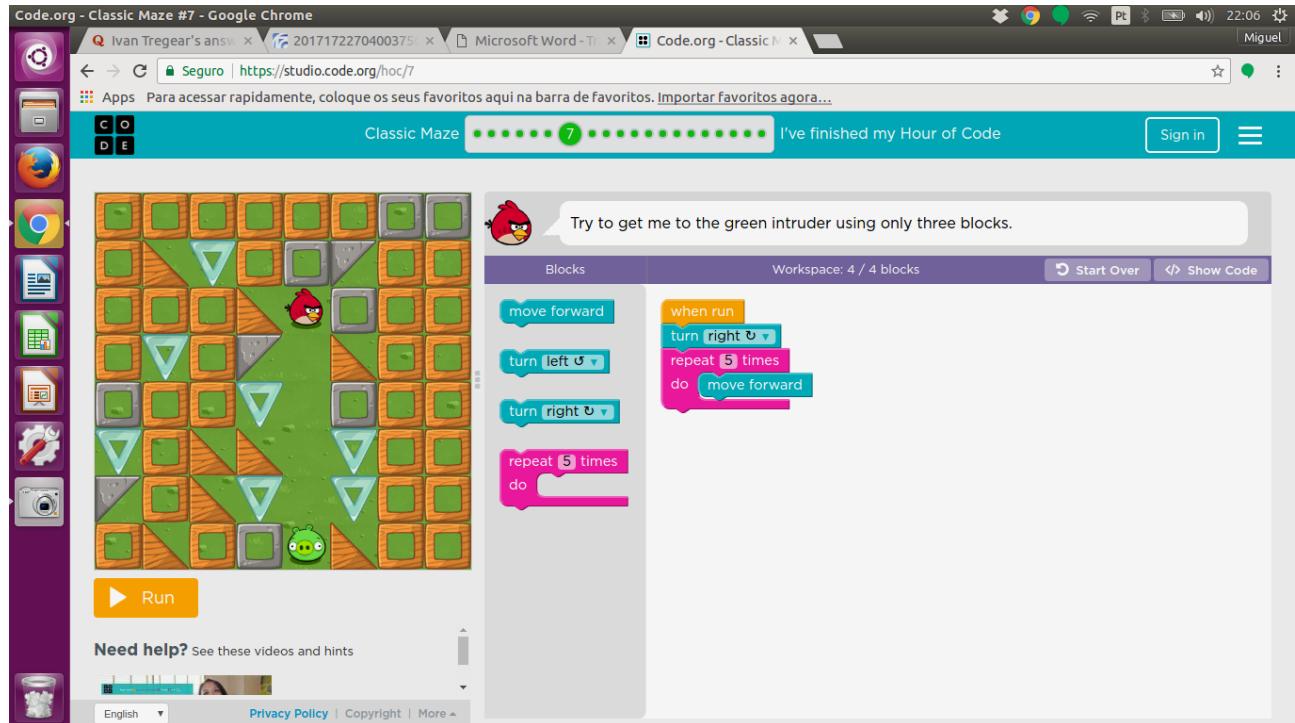
Código Fonte em Java Script:

```
for (var count = 0; count < 5; count++) {  
    moveForward();  
}
```

O algoritmo usa o comando “for” como laço, dentro das condições de for ele inicializa a variável count com 0, determina que count tem que ser menor que 5 e a cada giro ele adiciona 1 na variável count.

Como bloco de comando do laço “for” ele usa o “moveForward” , ou seja, o comando será executado 5 vezes, partindo de 0 e indo até 4, finalizando o algoritmo quando count = 5.

## DESAFIO 7.

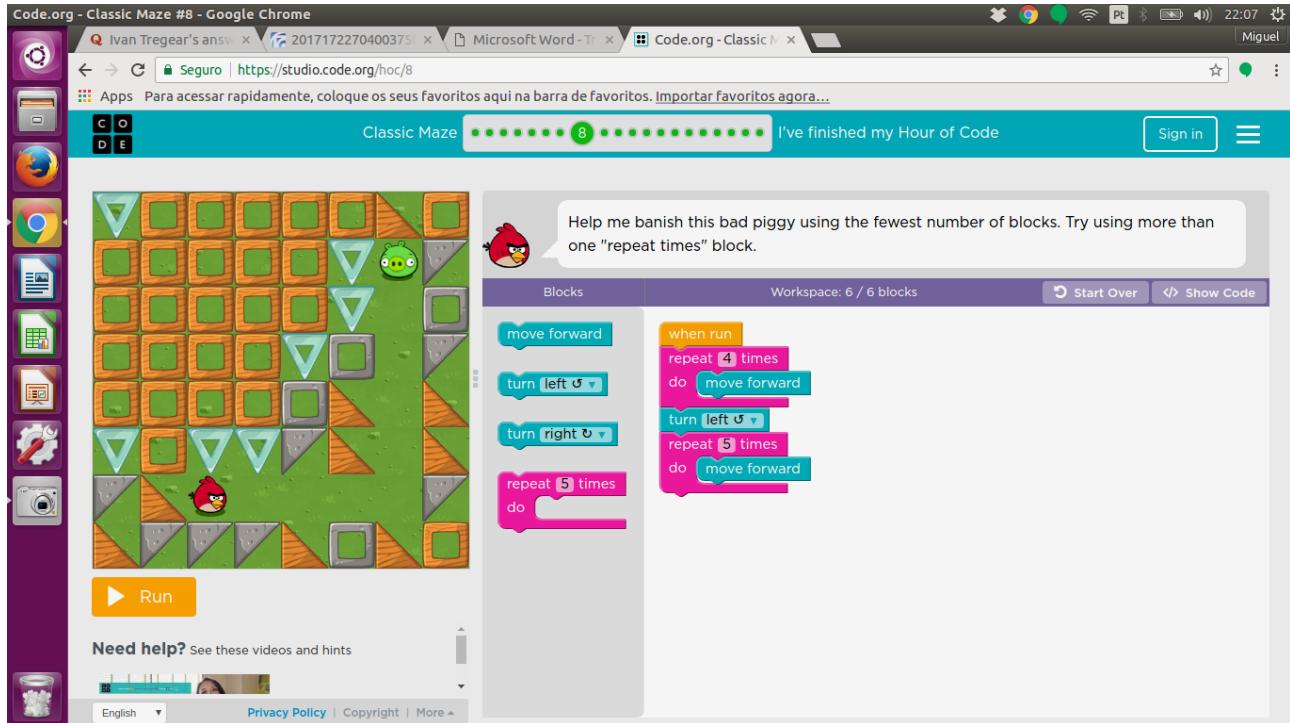


Analisando o desafio, observei que o pássaro estava direcionado para o lado errado, então determinei que ele deveria virar para a direção correta antes de começar a se mover. E usei novamente o “repeat” para fazer os movimentos repetitivos.

Código fonte em Java Script:

```
turnRight();  
  
for (var count = 0; count < 5; count ++){  
  
    moveForward();  
  
}  
  
Devido a necessidade de direcionar para o lado certo, o algoritmo colocando um comando para virar antes do laço, logo depois, utilizou o laço “for” com as mesmas condições usadas no desafio anterior.
```

## DESAFIO 8.



Analisando o desafio, observei que o pássaro deveria fazer vários movimentos repetitivos, porém em direções distintas.

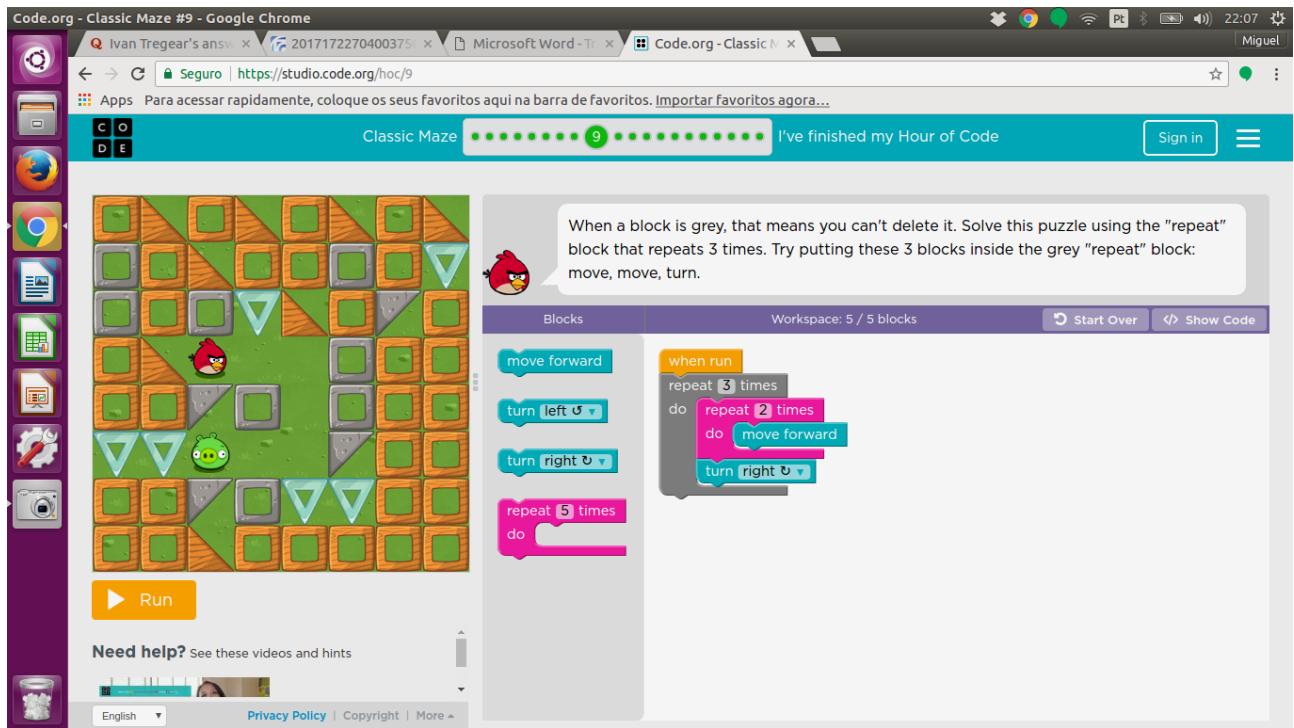
Primeiramente, utilizei um “repeat” com a condição que ele repita o comando dentro do laço 4 vezes, que no caso é mover-se. Logo depois, usei o “turnLeft” para direcionar o pássaro de forma correta e apliquei o laço com a condição de repetição igual a 5 vezes o comando “moveForward”.

Código fonte em Java Script:

```
for (var count = 0; count < 4; count ++){  
    moveForward( );  
}  
  
turnLeft( );  
  
for (var count = 0; count < 5; count ++)[  
    moveForward( );  
}
```

O algoritmo usa o laço duas vezes, o primeiro com a condição dada ser menor que 4, ou seja, repetir o comando dentro do laço 4 vezes, e o segundo com a condição dada ser menor que 5 repetindo o comando dentro do segundo laço, 5 vezes. É usado o comando “turnLeft” para concertar a trajetória do pássaro.

## DESAFIO 9.



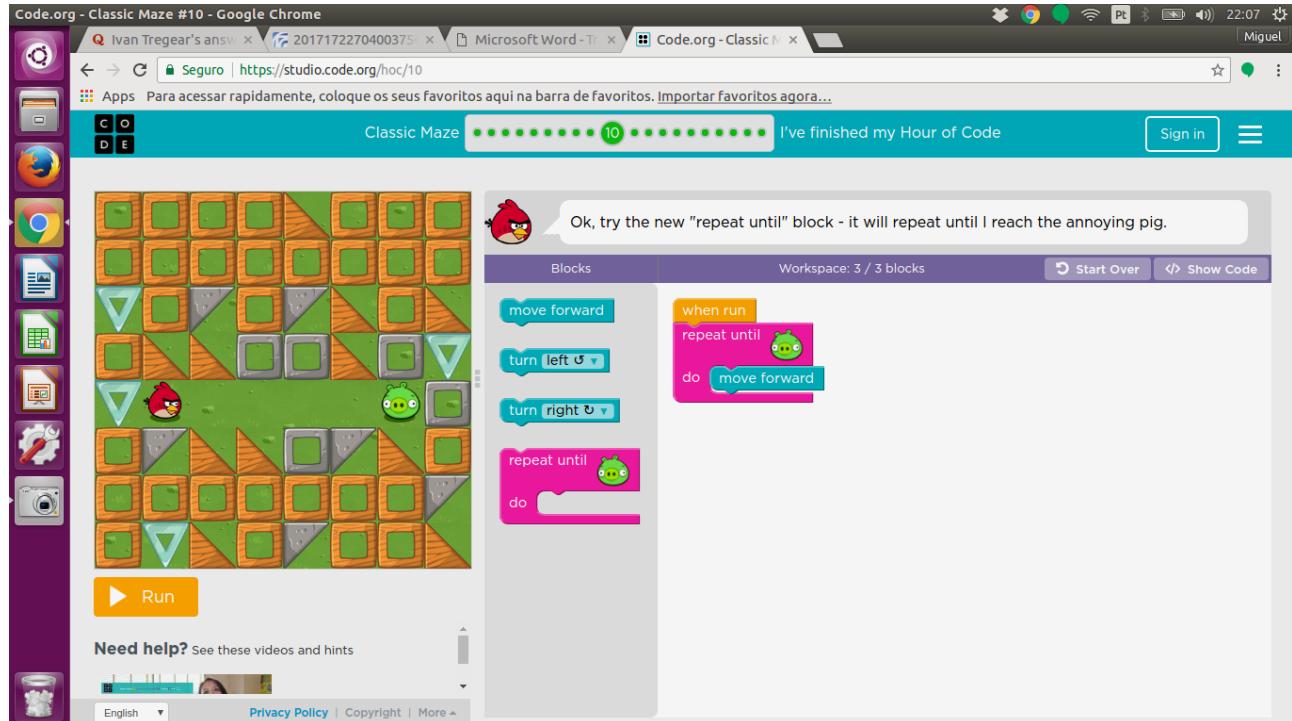
Analisando o desafio, ele propõe obrigatoriamente um “repeat” com condição de 3 giros no laço, e observando que o pássaro precisa repetir o comando “moveForward” 6 vezes, eu coloquei um “repeat” com condição de 2 giros e com o comando de mover dentro dele, ainda dentro do “repeat” externo eu coloquei o comando “turnRight”. A ordem de ação do pássaro começara com ele se movendo 2 vezes e depois virando pra direita, repetindo isso 3 vezes.

Código fonte em Java Script:

```
for (var count2 = 0; count2 < 3; count2++){
    for ( var count = 0; count < 2; count++){
        moveForward( );
    }
    turnRight( );
}
```

O código trabalha com um laço dentro de um laço, declarando duas variáveis para serem os contadores dentro do laço e inicializando ambas com 0, porém, com condições de parada diferentes, o laço externo utiliza 3 e o interno 2. O comando “turnRight” não pode ficar dentro do segundo laço, porque o pássaro viraria a cada avanço, e para o problema isso daria errado.

## DESAFIO 10.



Nesse desafio a condição de parada é somente quando chegar ao fim, ou seja, o pássaro chegar ao porco. Então o cuidado precisa ser tomado para que ele não realize movimentos cílicos sem que ele chegue num resultado “loop infinito”.

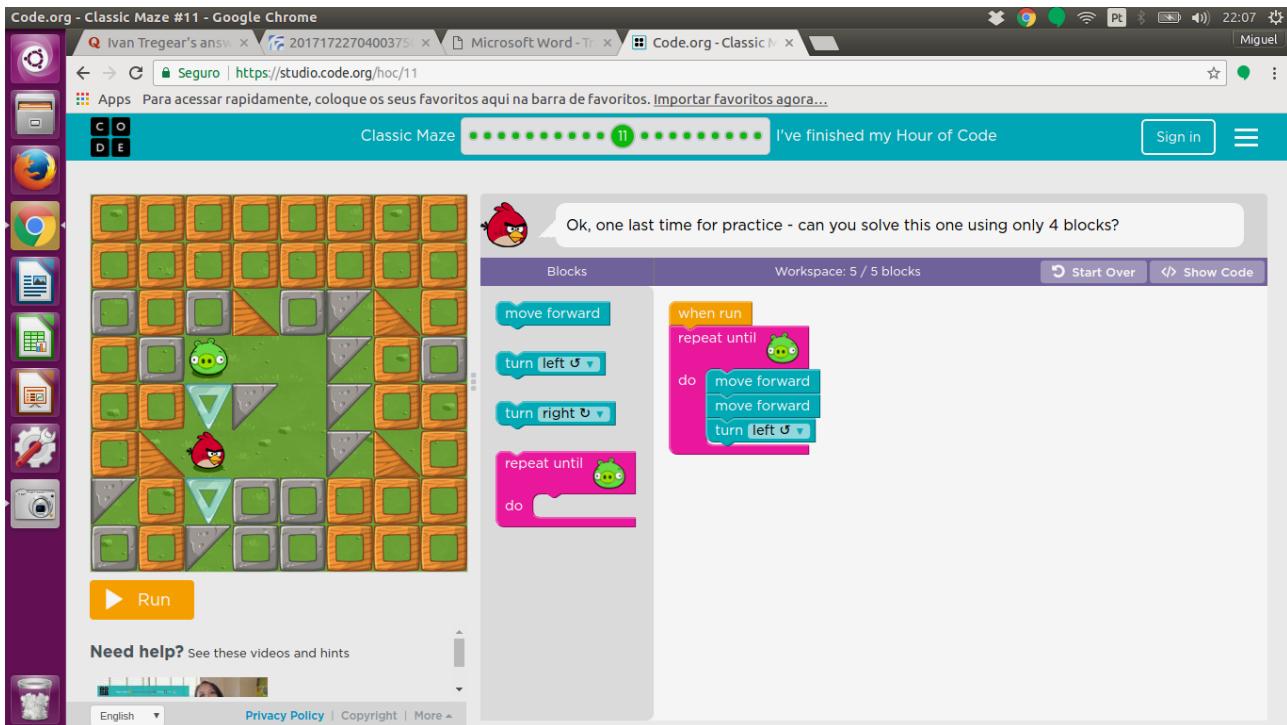
Como o pássaro está em linha reta com o porco o único comando dentro do “repeat” é “moveForward” até ele chegar ao porco.

Código fonte em Java Script:

```
while ( notFinished( ) ) {  
    moveForward( );  
}
```

Nesse algoritmo, o laço utilizado é o “while”, e a condição de parada é somente quando chegar ao fim, se os comandos não levarem a um fim, o algoritmo apresentará um erro.

## DESAFIO 11.



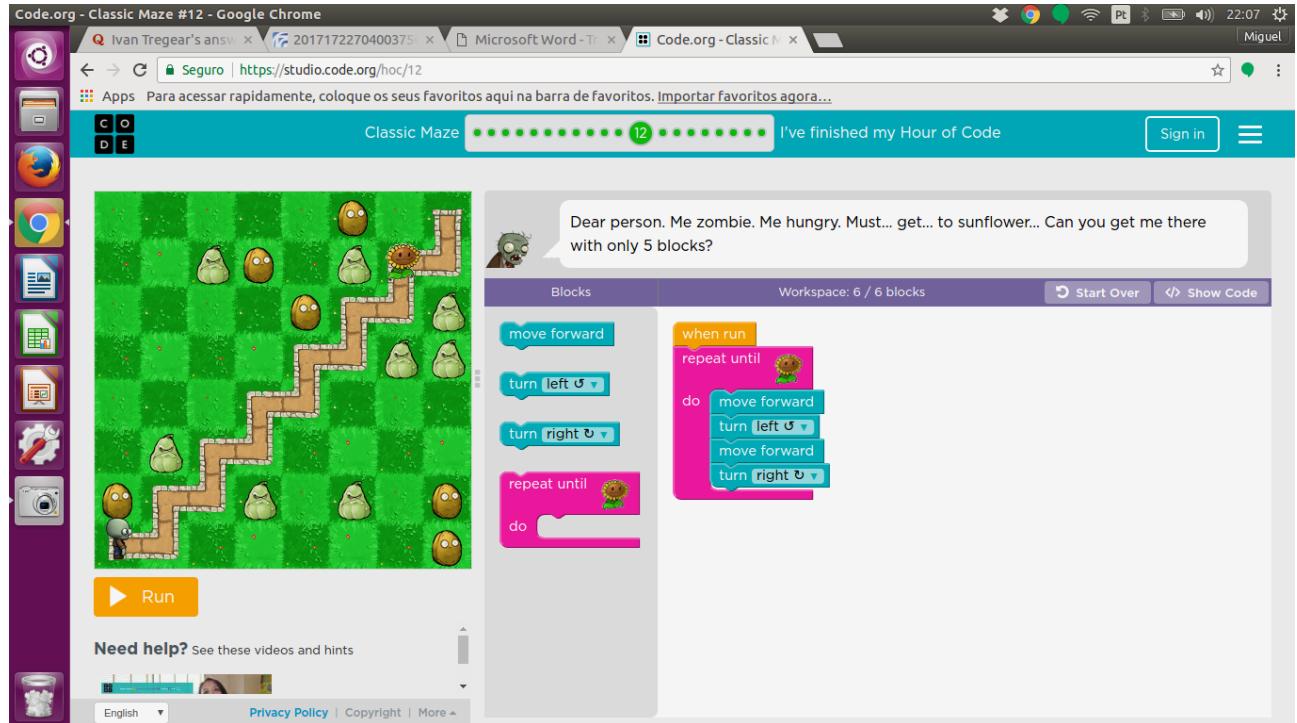
Analisando o desafio, observei que o pássaro precisaria fazer dois “moveForward” antes de virar a esquerda, então usando o “repeat” e levando em conta que o critério de parada é chegar ao fim, coloquei todos os movimentos repetitivos dentro do laço.

Código fonte em Java Script:

```
while ( notFinished ( ) ) {  
  
    moveForward ( );  
  
    moveForward ( );  
  
    turnLeft ( );  
  
}
```

O algoritmo utiliza o “while” e é inserido todas as ações repetitivas dentro dele, o critério de parada utilizado é chegar ao fim.

## DESAFIO 12.



Analisando o desafio, observei que o zumbi precisaria avançar, virar para esquerda avançar e virar pra direita, então coloquei todos esses comando dentro do “repeat” que tem a condição de parada quando chegar ao final.

Código fonte em Java Script:

```
while ( notFinished( ) ){
  moveForward( );
  turnLeft( );
  moveForward( );
  turnRight( );
}
```

O algoritmo trabalha com o “while” já visto antes, e com a repetição dos comandos dentro dele.

## DESAFIO 13.



Analisando o desafio, organizei os blocos de comando dentro do “repeat” de forma que o zumbi desvie da planta carnívora e chegue até a outra planta com êxito.

Código fonte em Java Script:

```
while ( notFinished( ) ) {  
    turnRight( );  
    moveForward( );  
    turnLeft( );  
    moveForward( );  
}
```

O algoritmo é organizado ainda com o “while” e com os comandos dentro do bloco de comandos repetindo.

## DESAFIO 14.

The screenshot shows a browser window with the Code.org website open. The title bar indicates it's running in Google Chrome. The main content is a 'Classic Maze' challenge titled '14'. On the left, there's a green grassy field with several zombie icons scattered around. A single zombie is at the bottom left, facing right. To its right is a vertical path made of brown blocks. At the top of this path is another zombie. The goal is to get the zombie to the top. In the workspace on the right, there are several Scratch-style blocks:

- A yellow 'when run' hat block.
- An orange 'repeat until' control block attached to a grey 'move forward' movement block.
- A grey 'do' control block attached to the 'repeat until' loop.
- A blue 'if path to the left' control block attached to a grey 'turn left' movement block.
- A grey 'do' control block attached to the 'if path to the left' block.

A speech bubble from a zombie says: "Use the new 'if' block to let me decide when to turn. Hint: you only need one more block, but learn how we set it up so you can do it on your own next time."

Nesse desafio trabalha com a condicional “if” que fica a todo momento buscando quando será verdade e realizar o bloco de comandos dentro dela.

Nesse desafio, o zumbi avança até chegar o momento que ter um caminho a esquerda se torna verdade, ai ele realiza o bloco de comandos dentro do “if” e volta a avançar, até chegar a planta.

Código fonte em Java Script:

```
while ( notFinished( ) ) {  
    moveForward( );  
    if ( isPathLeft ( ) ){  
        turnLeft( );  
    }  
}
```

Nesse algoritmo, é usado também o comando “if” de condicional, que serve para comparar e nesse caso, quando há o espaço para ele virar a esquerda, esse ativa o bloco de comandos dentro dele e vira.

## DESAFIO 15.

Code.org - Classic Maze #15 - Google Chrome  
Ivan Tregebar's ans... | Seguro | https://studio.code.org/hoc/15 | Microsoft Word - T... | Code.org - Classic | 22:08 Miguel

Classic Maze I've finished my Hour of Code

Blocks Workspace: 5 / 5 blocks Start Over Show Code

move forward  
turn left  
turn right  
repeat until  
do  
if path to the right  
do

Ok, this is just like the last puzzle, but you need to remember how you used the "if" block and the "repeat" block together.

Run Need help? See these videos and hints English Privacy Policy Copyright More

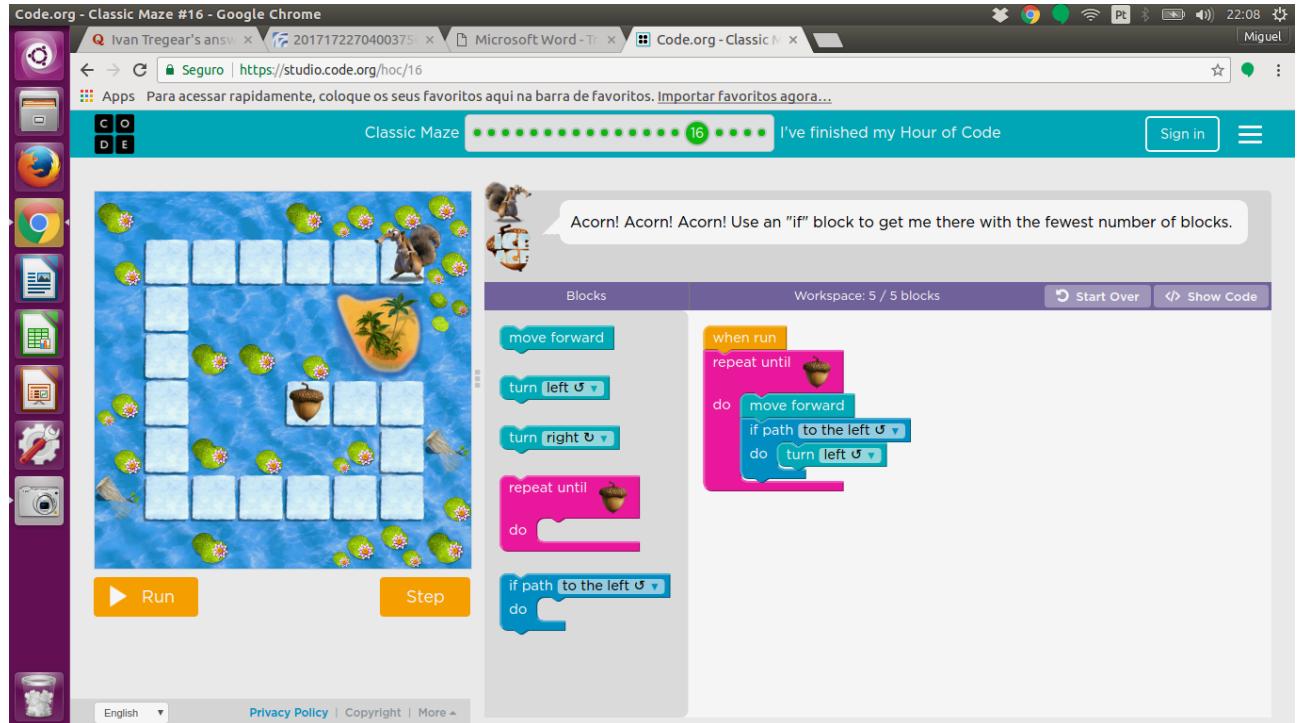
Nesse desafio usasse novamente o laço em conjunto com a condiciona “se” , porém agora ele compara para a direita, e quando há estrada para a direita, ele vira. Mesmo tendo caminho pros outros lados, ele os ignora.

Código fonte em Java Script:

```
while ( notFinished( ) ) {  
    moveForward( );  
  
    if ( isPathRight( ) ) {  
        turnRight( );  
    }  
}
```

Nesse algoritmo ele usa a mesma organização e comandos que o do desafio anterior, com a mecânica de comparação e sempre isso acontecendo até chegar ao resultado.

## DESAFIO 16.



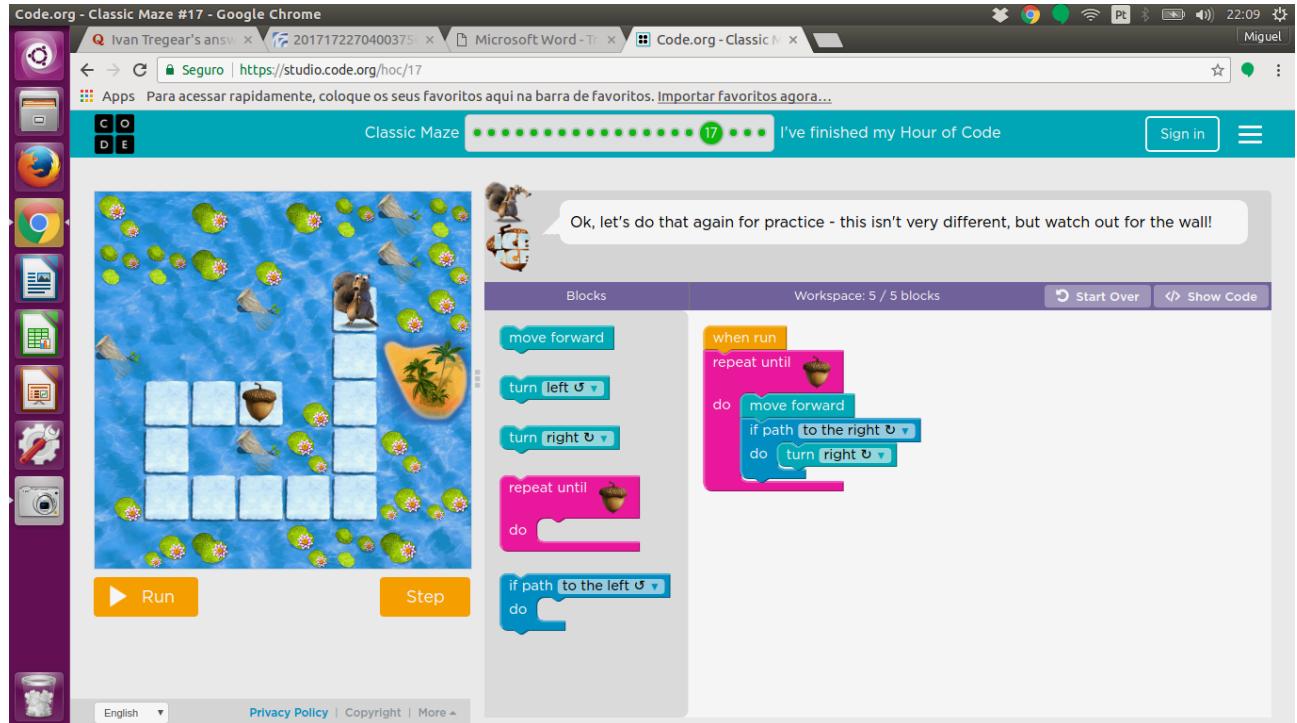
Nesse novamente usa o laço com condição de parada quando chegar a noz, dentro desse laço usasse a condicional que a cada passo testa se tem caminho a esquerda e quando isso é verdade ele vira e continua o avanço.

Código fonte em Java Script:

```
while ( notFinished () ) {  
    moveForward();  
  
    if ( isPathLeft() ) {  
        turnLeft();  
  
    }  
}
```

O algoritmo utiliza o mesmo sistema do anterior, uma condicional dentro de um laço, executando a bloco de comandos da condicional somente quando ela for verdadeira.

## DESAFIO 17.



O desafio é equivalente ao anterior, porém com a condicional para o outro lado, com a condicional considerando o caminho à direita.

Código fonte em Java Script:

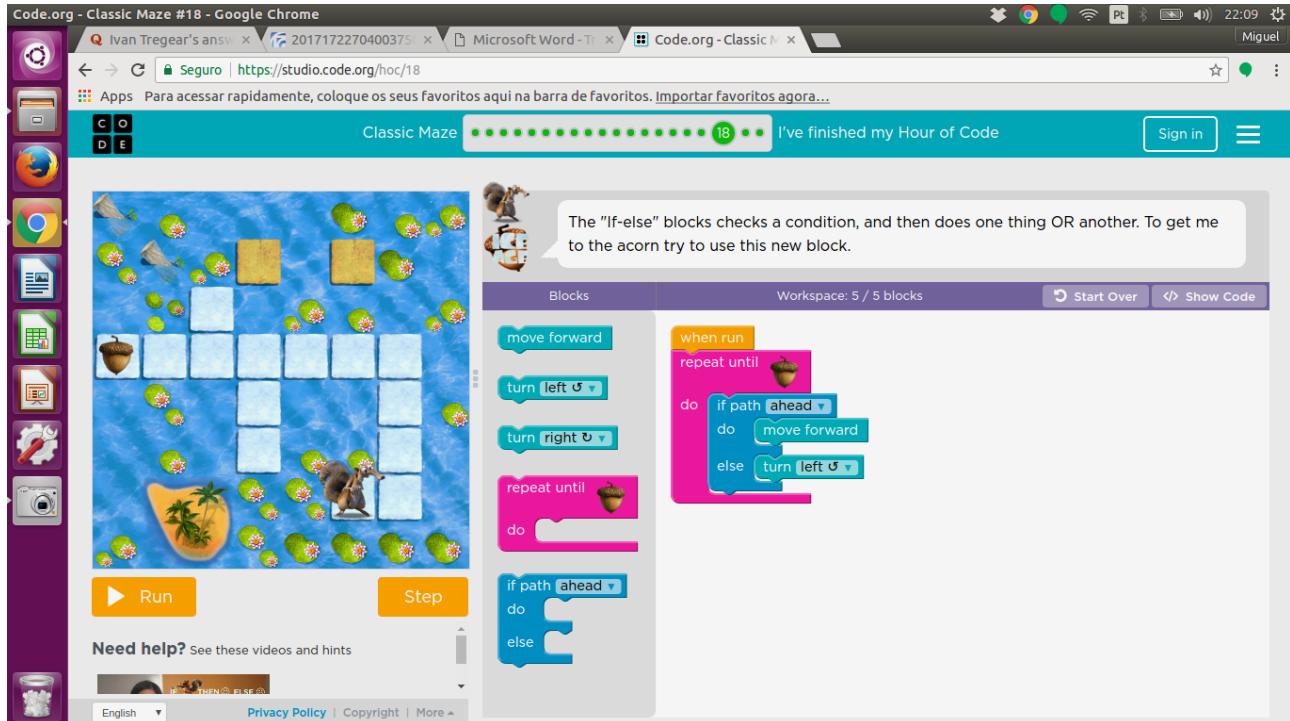
```
while ( notFinished () ){

    moveForward();

    if (isPathRight() ) {
        turnRight();
    }
}
```

O algoritmo é organizado da mesma forma e com os mesmo comandos, porém com a condicional trabalho com o outro lado.

## DESAFIO 18.



Analisando o desafio, e tendo um novo comando, o “ if e else” ele realiza uma ação quando a condição de “if” for verdadeira e outra pra quando for falsa, executa o bloco dentro de “else”.

Nesse caso ele compara se há caminho a frente, sempre que for verdade ele irá executar a ação de avançar, quando não for, ele irá virar pra esquerda.

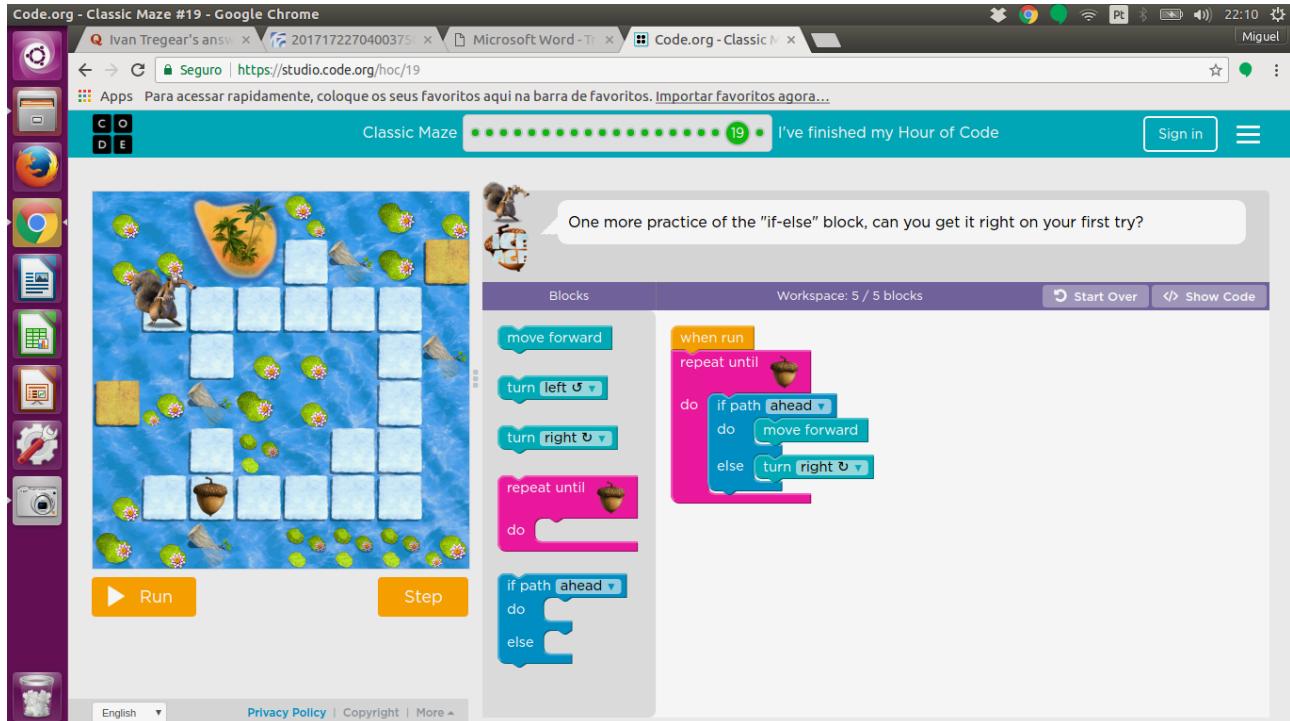
Código fonte em Java Script:

```
while ( notFinished ( ) ) {  
  
    if ( isPathForward ( ) ){  
  
        moveForward( );  
  
    } else {  
  
        turnLeft( );  
    }  
}
```

Nesse algoritmo aparece o “else” como novo, ele é executado quando a comparação de “if” for falsa.

Então utilizando o “if e else” você realiza uma comparação com a possibilidade de obter resultados diferentes, se verdadeira ou não.

## DESAFIO 19.



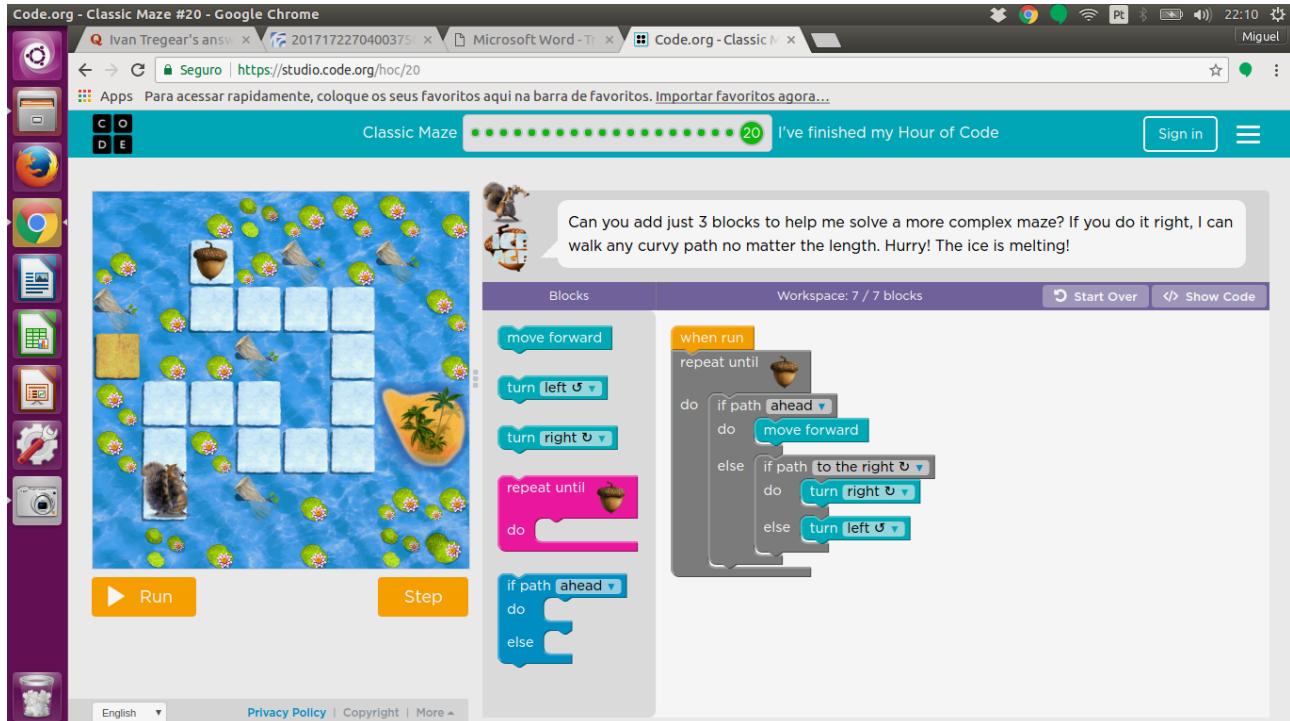
Analisando o desafio, os comandos são organizados da mesma forma que o desafio anterior, porém pro outro lado, realizando a comparação quando verdadeira executa um bloco e quando falsa o outro.

Código Fonte em Java Script:

```
while ( notFinished () ) {  
    if ( isPathForward( ) ){  
        moveForward ( );  
    } else {  
        turnRight ( );  
    }  
}
```

O algoritmo usa novamente o sistema de “if e else” dentro do laço, porém agora o bloco de comandos dentro do “else” é virar para a direita.

## DESAFIO 20.



Nesse desafio é determinado que terá de ser usado um laço, e uma condicional dentro de uma condicional.

A primeira condicional compará se há comando a frente , se sim, o esquilo avança, se não, executa a segunda condicional que compara se há caminho para a direita, se sim, ele vira para a direita, se não , ele vira para a esquerda e continua avançando e repetindo essas condições até chegar na noz.

Código fonte em JavaScript:

```
while ( notFinished() ) {
  if ( isPathForward() ){
    moveForward();
  } else {
    if ( isPathRight() ){
      turnRight();
    } else {
      turnLeft();
    }
  }
}
```

Nesse algoritmo é usado o “ Se senão se senão” para fazer em sequênci as comparações e executando o bloco de comando respectivo, quando verdadeiro.