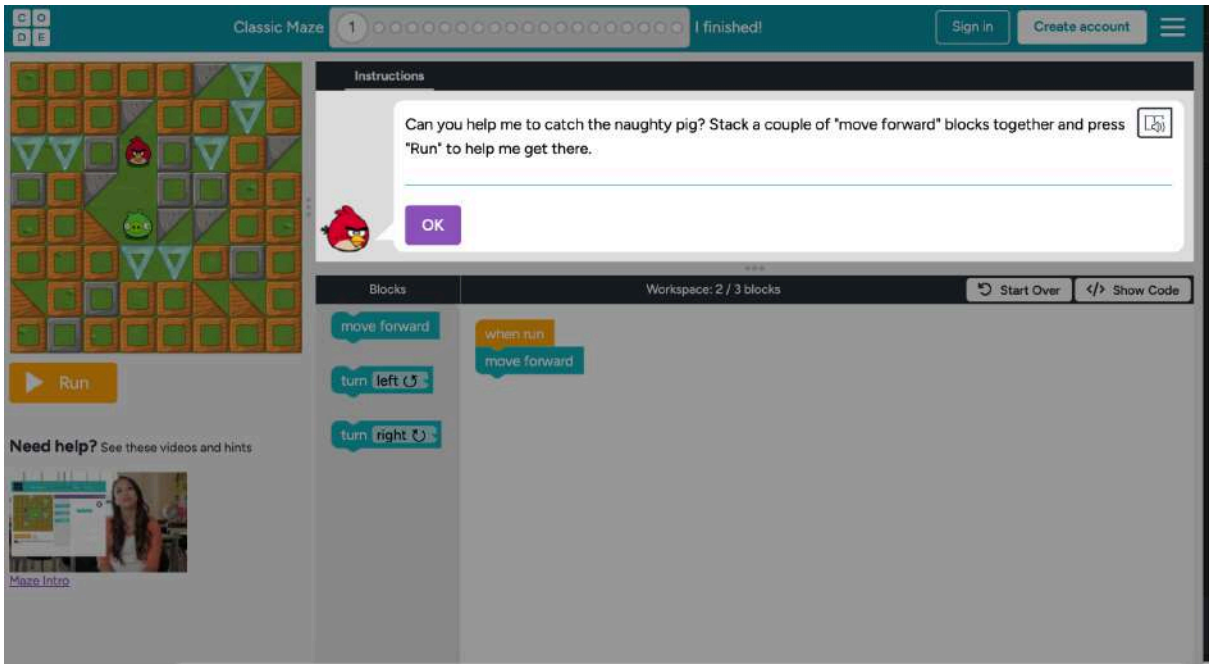
TP “Angry birds”

Sommaire

1. Introduction

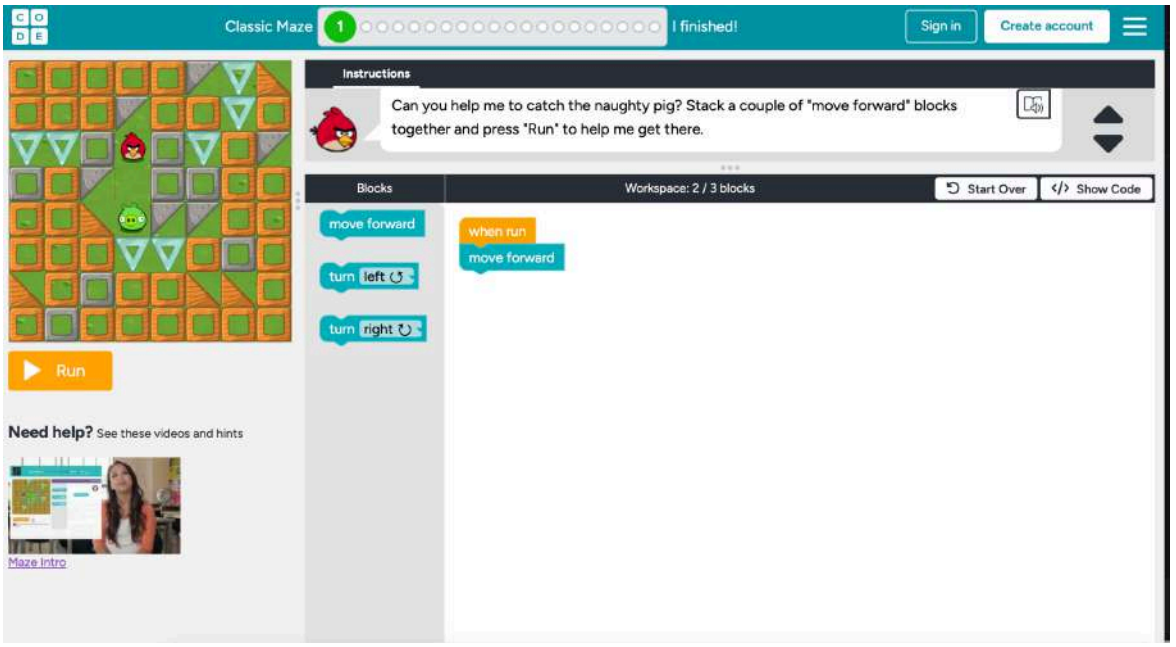
Dans ce TP nous allons apprendre la programmation par blocs.



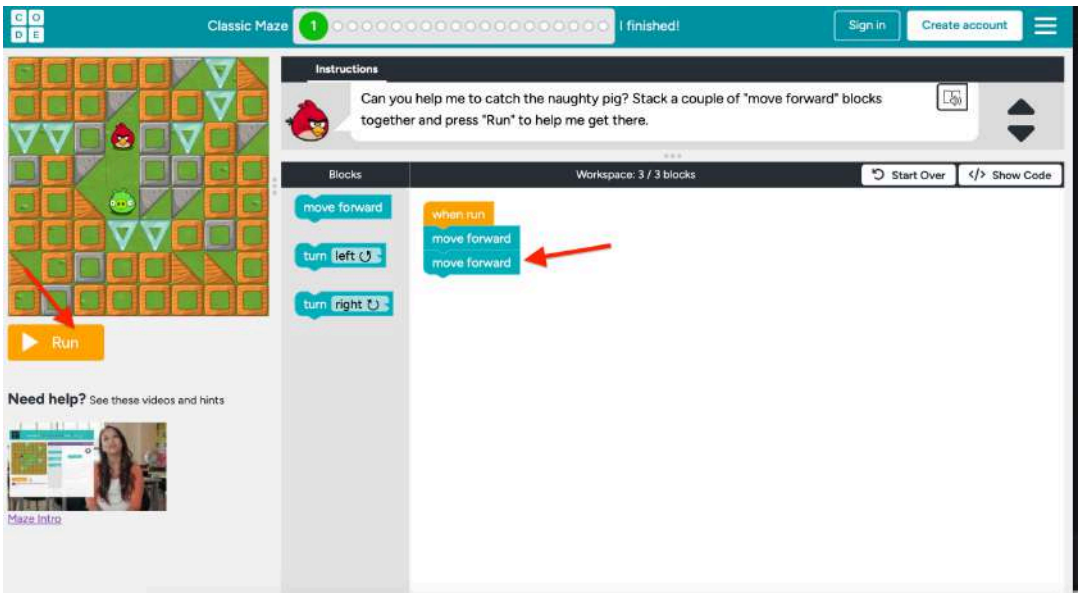
Etape 1

Le but est que le petit oiseau attrape le cochon, nous pouvons voir que l'oiseau rouge est situé 2 cases au-dessus du cochon.

Si je clique sur "run" il ne se déplacera que de seulement 1 case étant donné qu'il y a qu'une seule instruction "move forward"

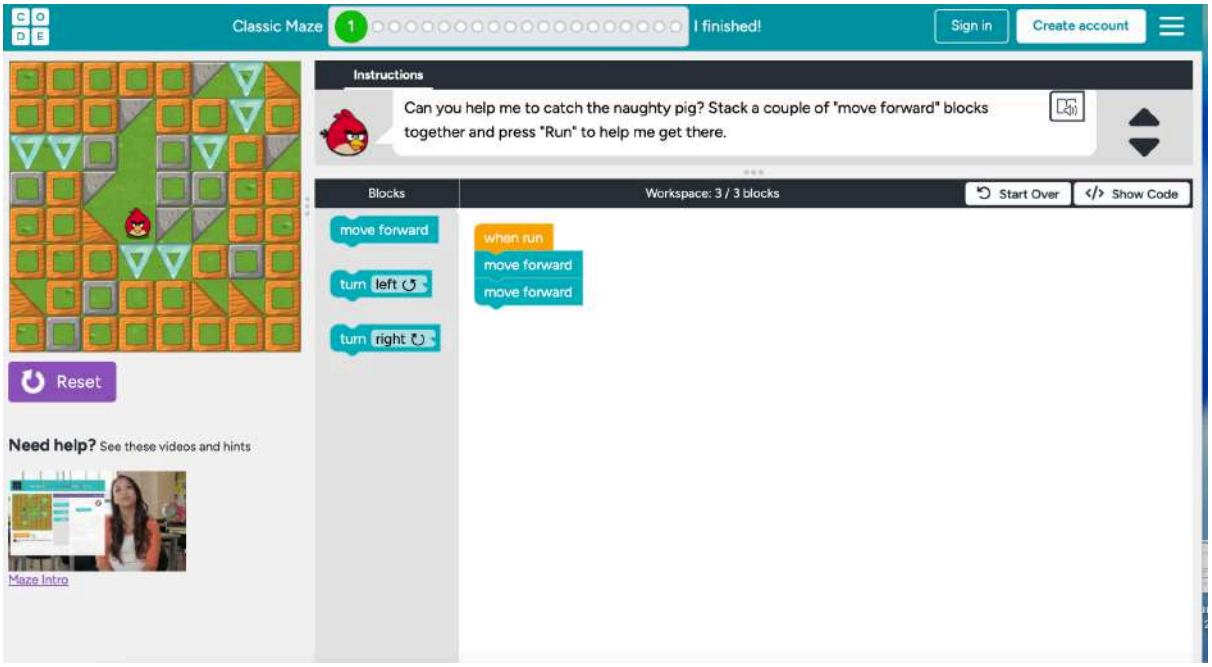


Pour le faire avancer de 2 cases il faut donc ajouter une deuxième instruction "move forward"

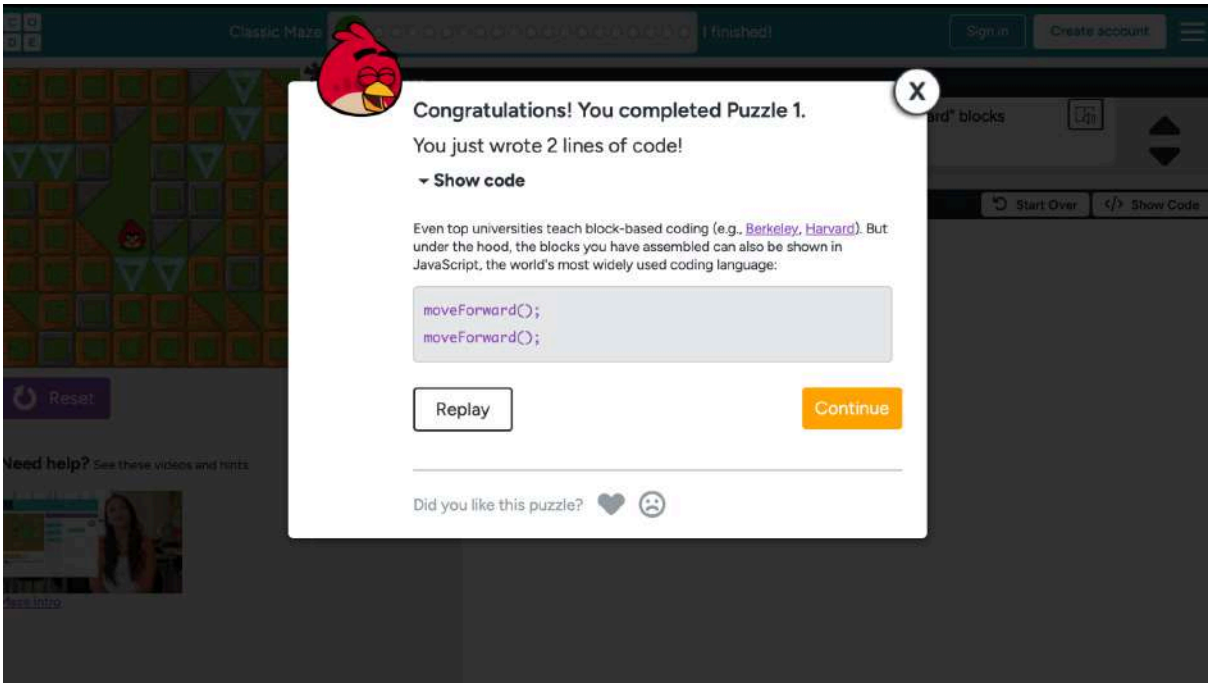


Etape 1.1

Une fois cette deuxième instruction ajoutée, l'oiseau a pu rejoindre sa destination.

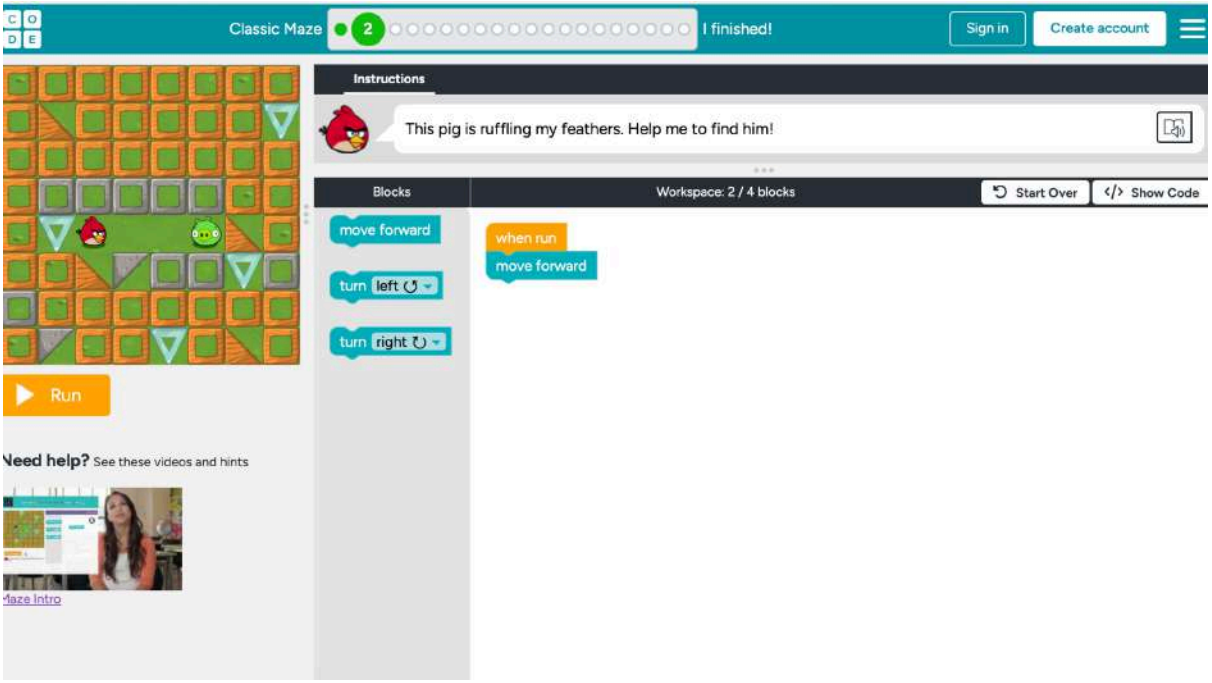


moveForward(); est la ligne de code qui indique à l'ordinateur de faire quelque chose.

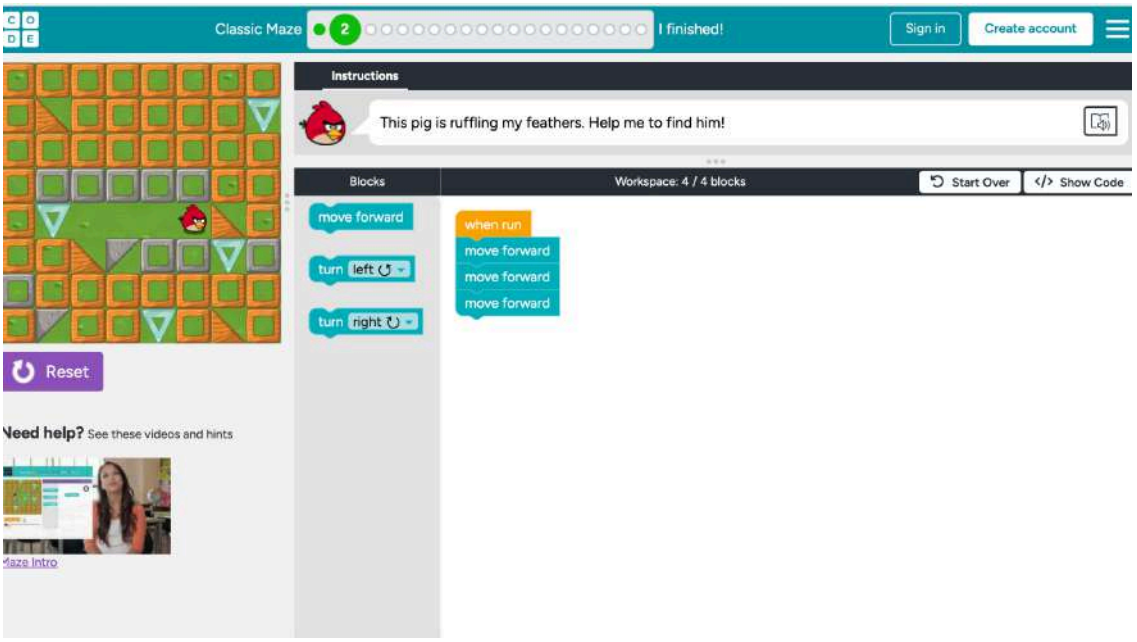


Etape 2

Même cas de figure, cette fois il y a 3 cases

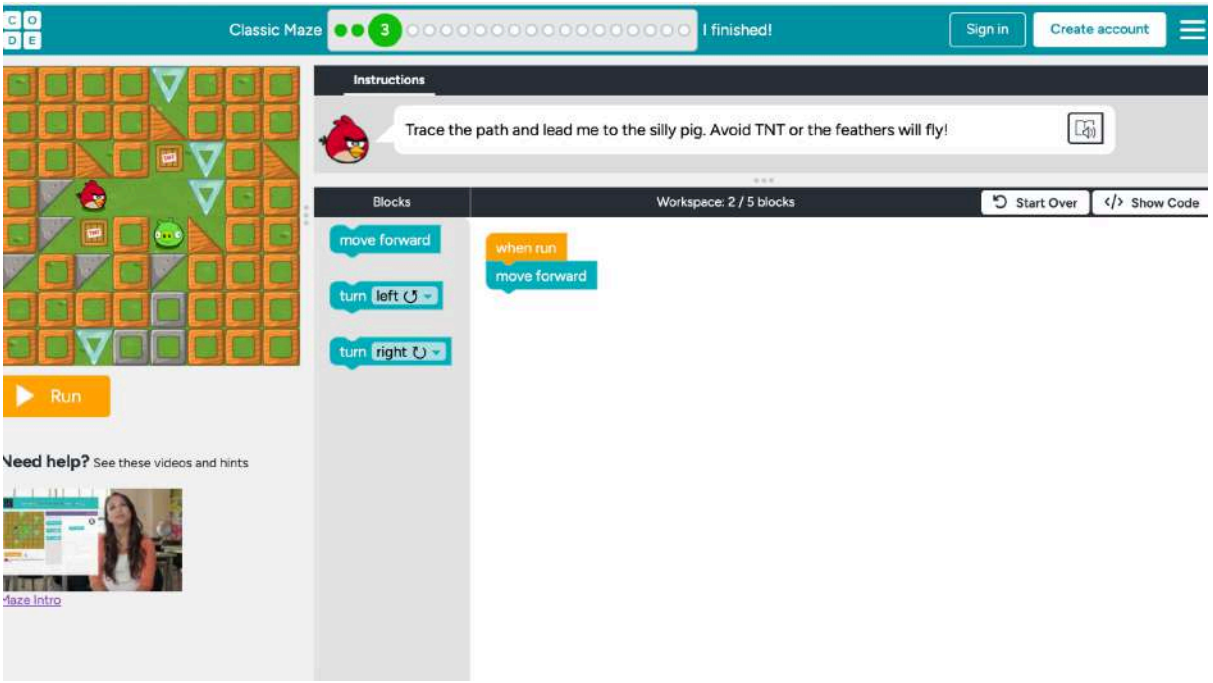


Nous allons donc lui donner 3 instructions “Move forward”

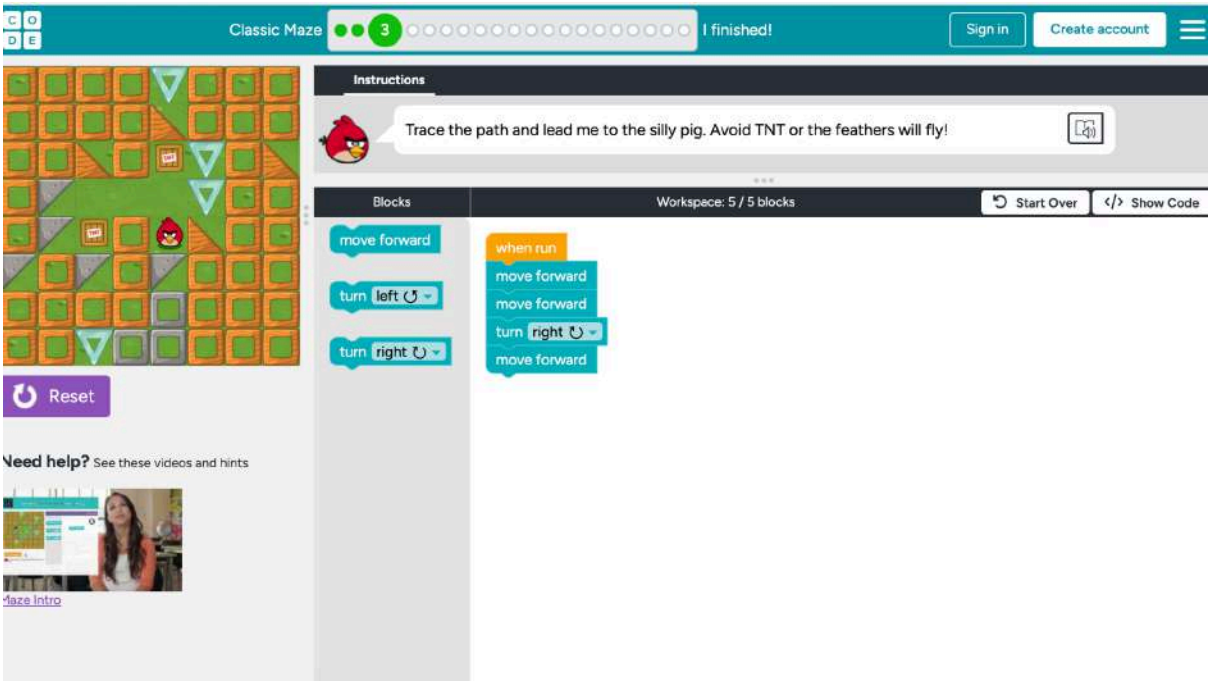


Etape 3

Cette étape demande une nouvelle instruction, nous allons devoir faire pivoter l'oiseau afin qu'il atteigne son objectif tout en prenant le soin d'éviter les blocs de TNT.

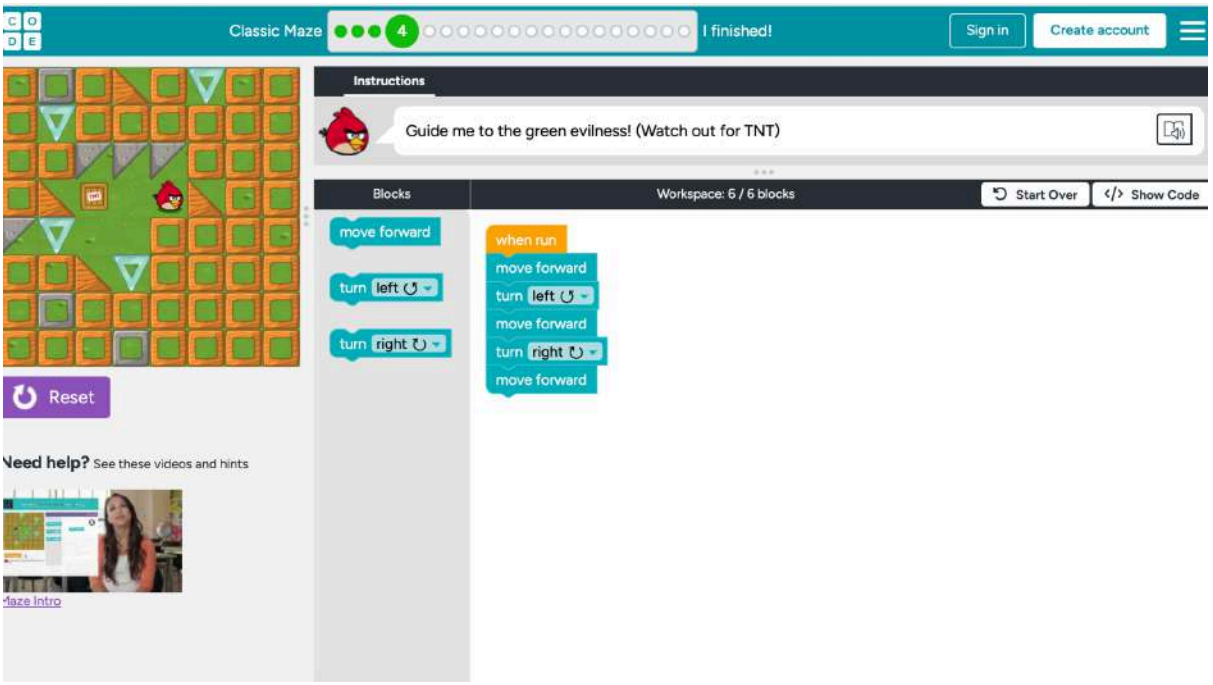


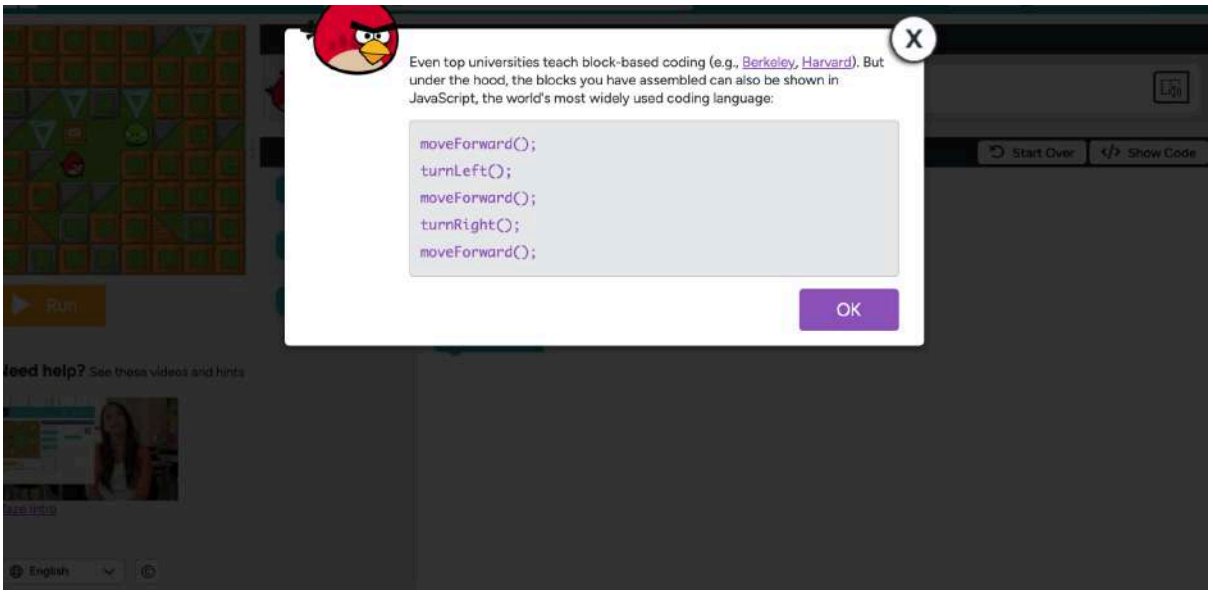
Nous allons ajouter moveForward(); afin de le mettre dans la bonne direction.



Etape 4

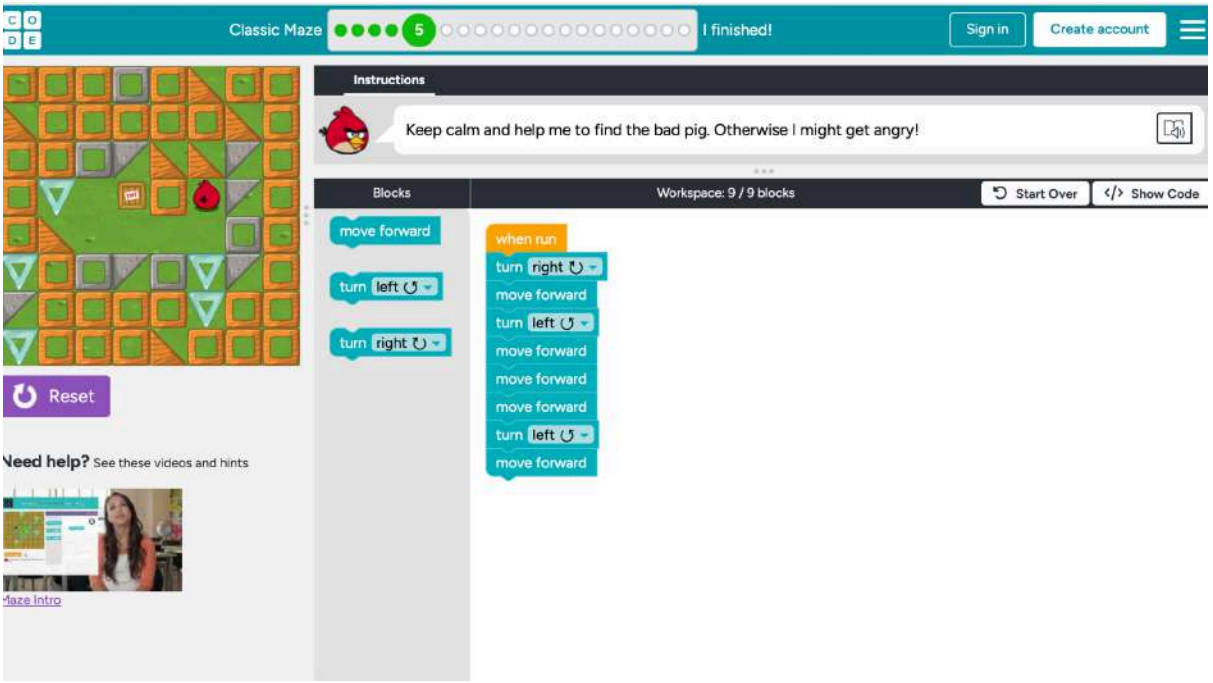
L'étape 4 nécessite une nouvelle instruction, celle permettant de pivoter à gauche qui se traduit par "TurnLeft();"



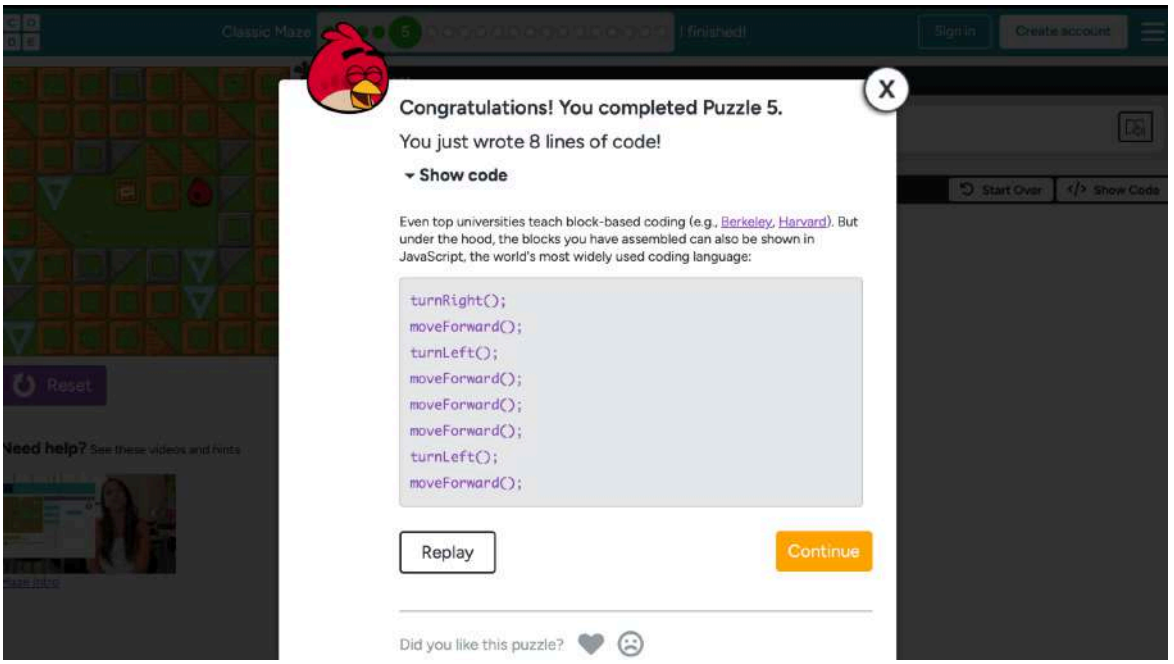


Etape 5

L’étape 5 commence à demander de plus en plus de lignes de code.



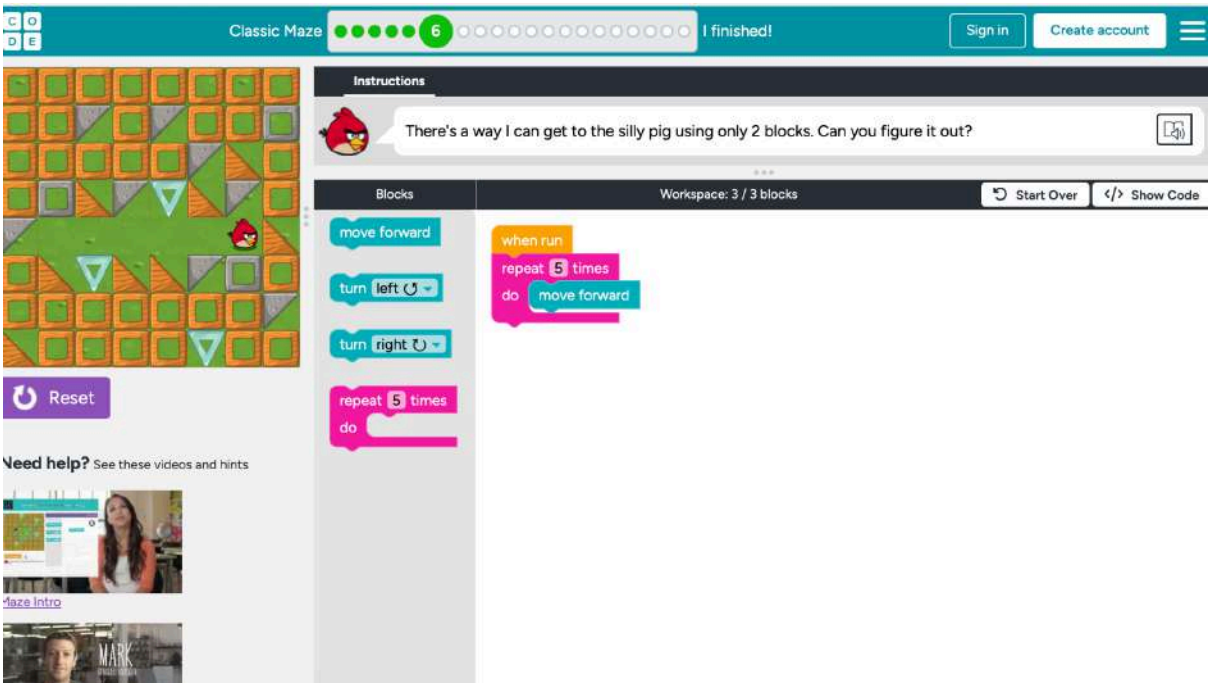
Nous en sommes à 8 lignes.



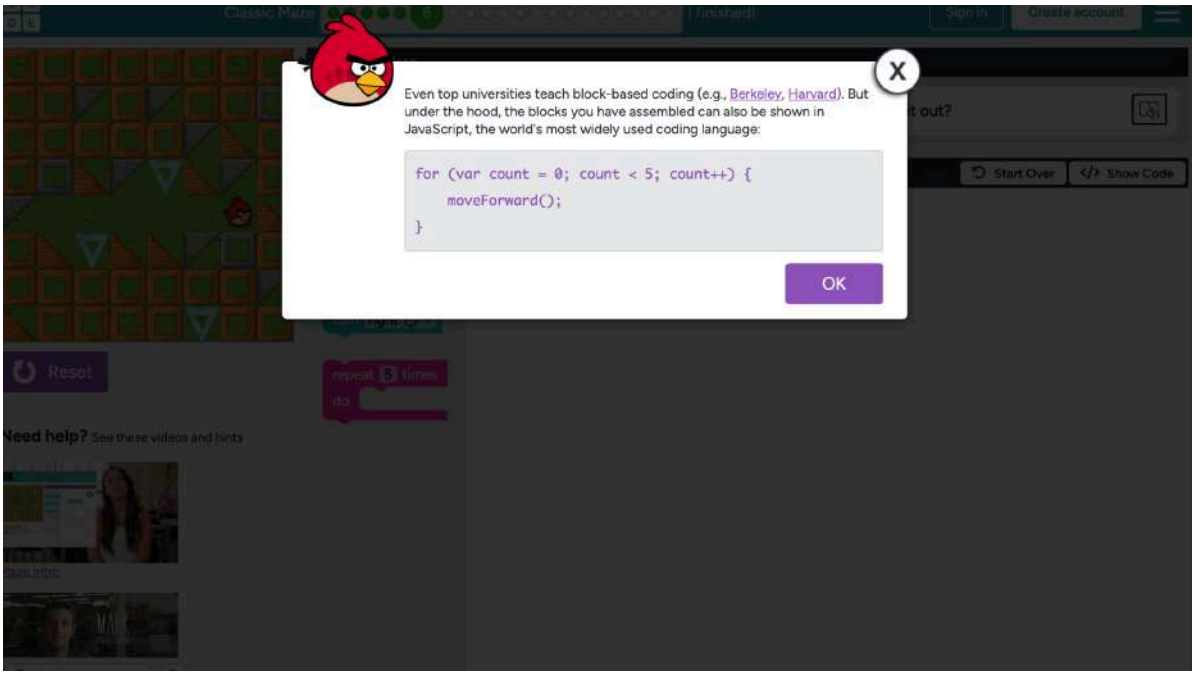
Etape 6

Nous allons maintenant découvrir les “boucles” La boucle dans ce programme permet de répéter l'instruction `moveForward();` 5 fois.

Cela permet de faire avancer le personnage 5 pas sans écrire l'instruction plusieurs fois.

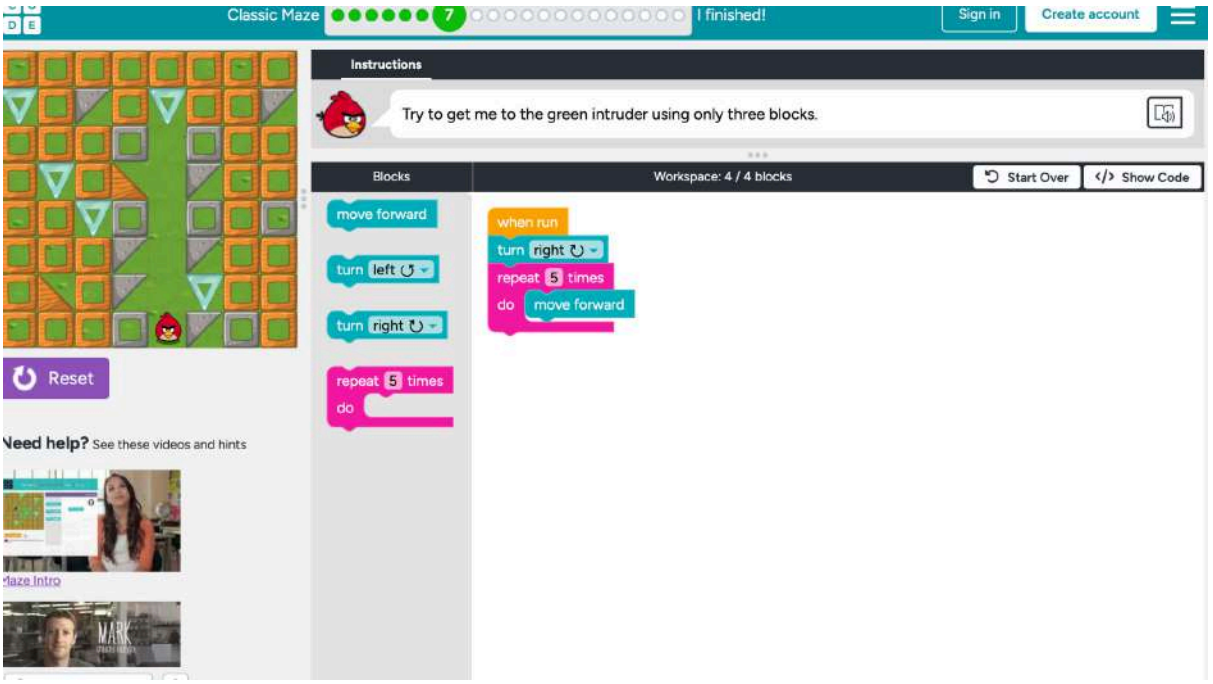


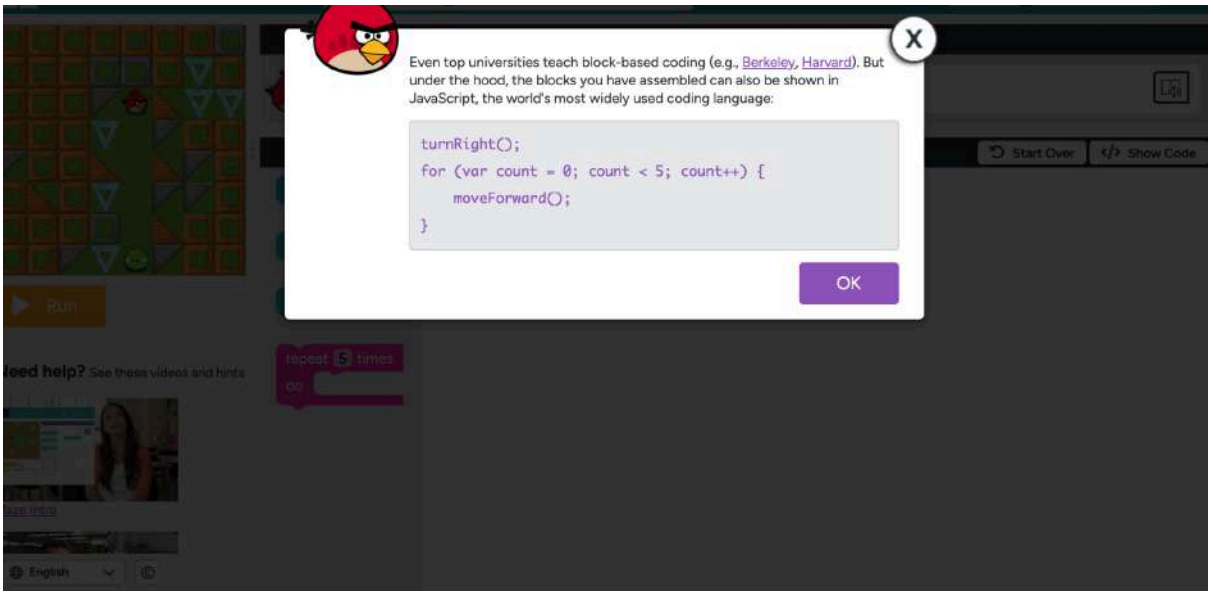
Voici comment cette boucle s’écrit.



Etape 7

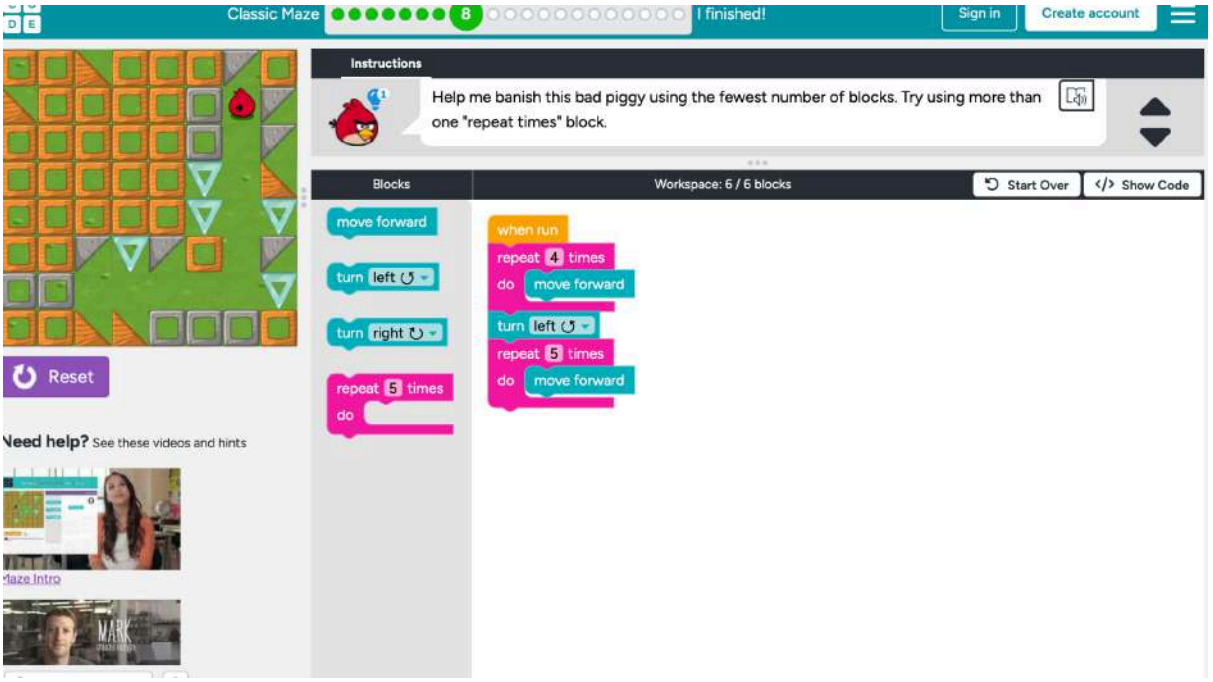
Dans l'étape 7, nous voulons que l'oiseau ne tourne qu'une seule fois donc nous commençons par écrire turnRight(), ensuite, la boucle for(var count = 0; count <5; count+++ va répéter ce qui est à l'intérieur 5 fois, on ajoute dedans moveForward() pour qu'il avance 5 fois.

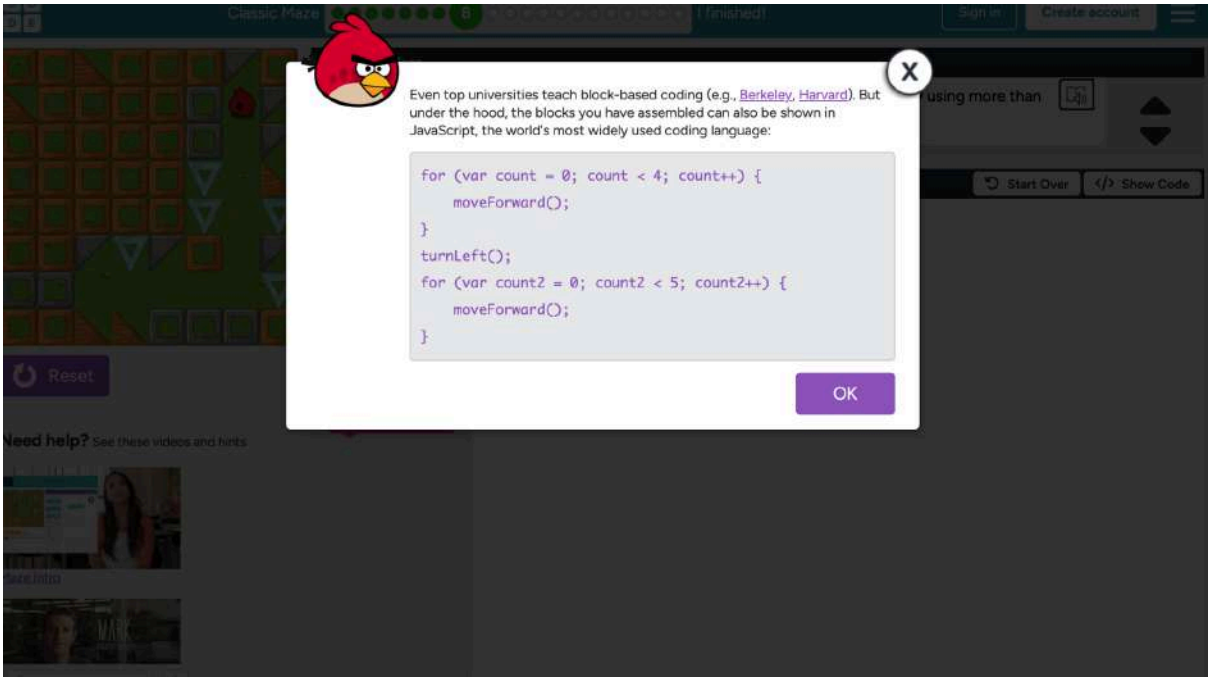




Etape 8

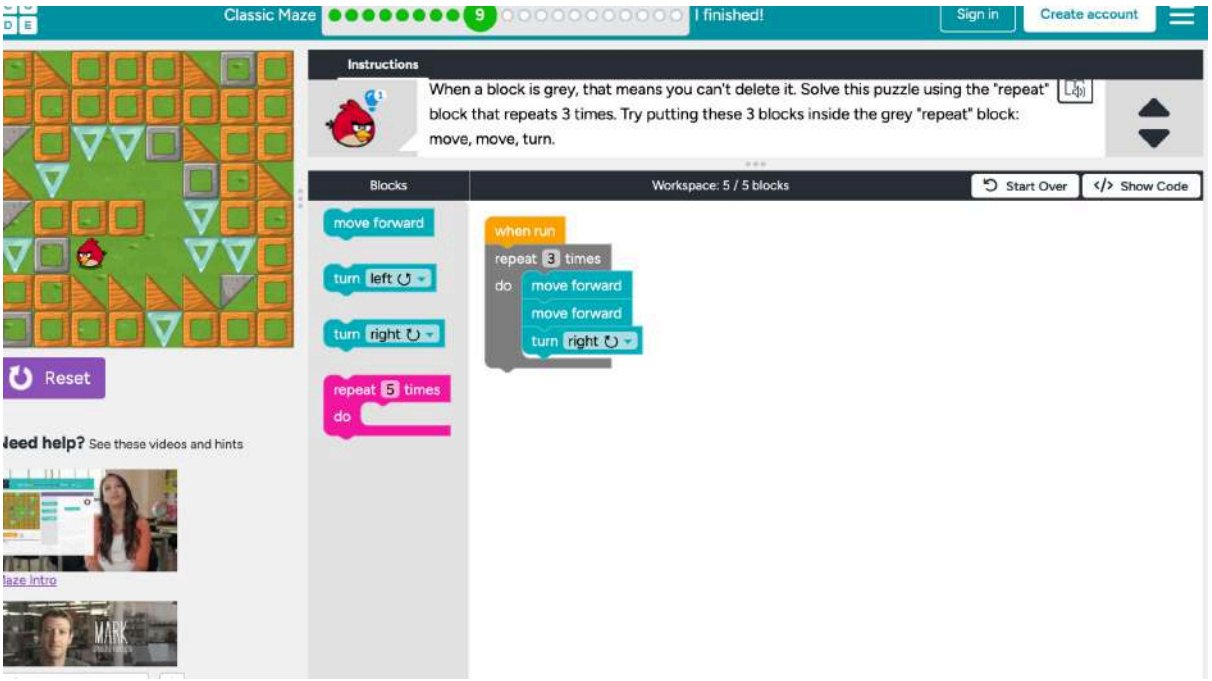
Dans l'étape 8, nous utilisons 2 blocs de répétitions, la première en est dotée de 4.
La boucle for (var count = 0; count <4; count +++) → l'oiseau avance 4 fois.
turnLeft() → il tourne à gauche.
for (var count2 = 0; count2 <5; count2++) → il avance encore 5 fois.

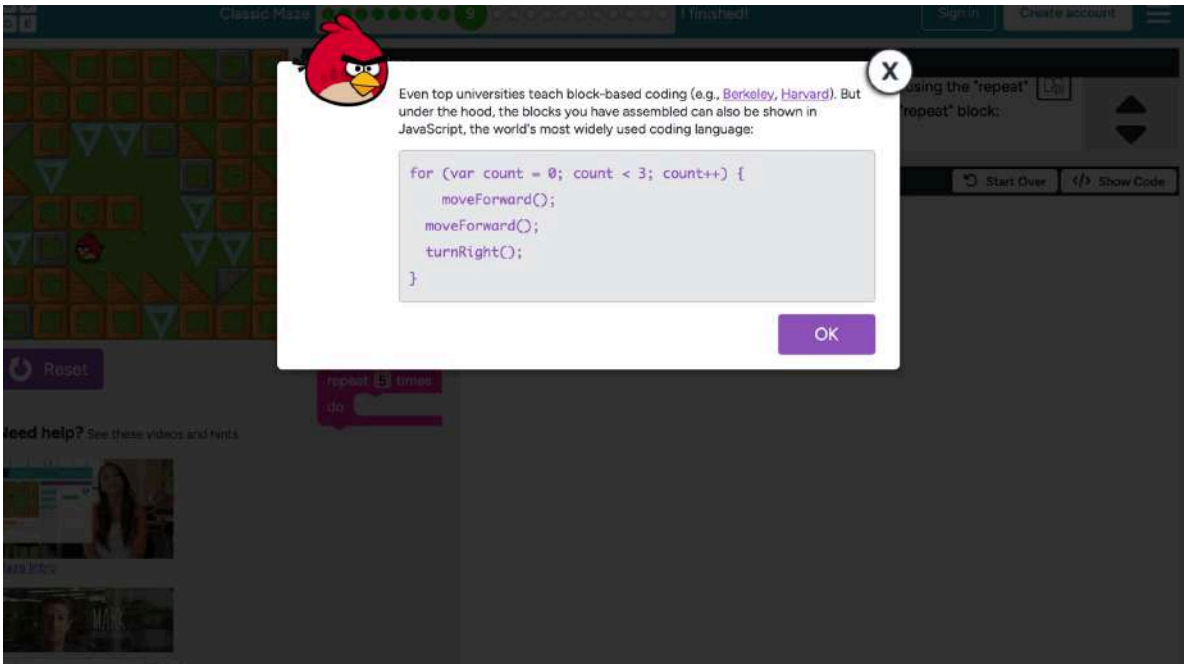




Etape 9

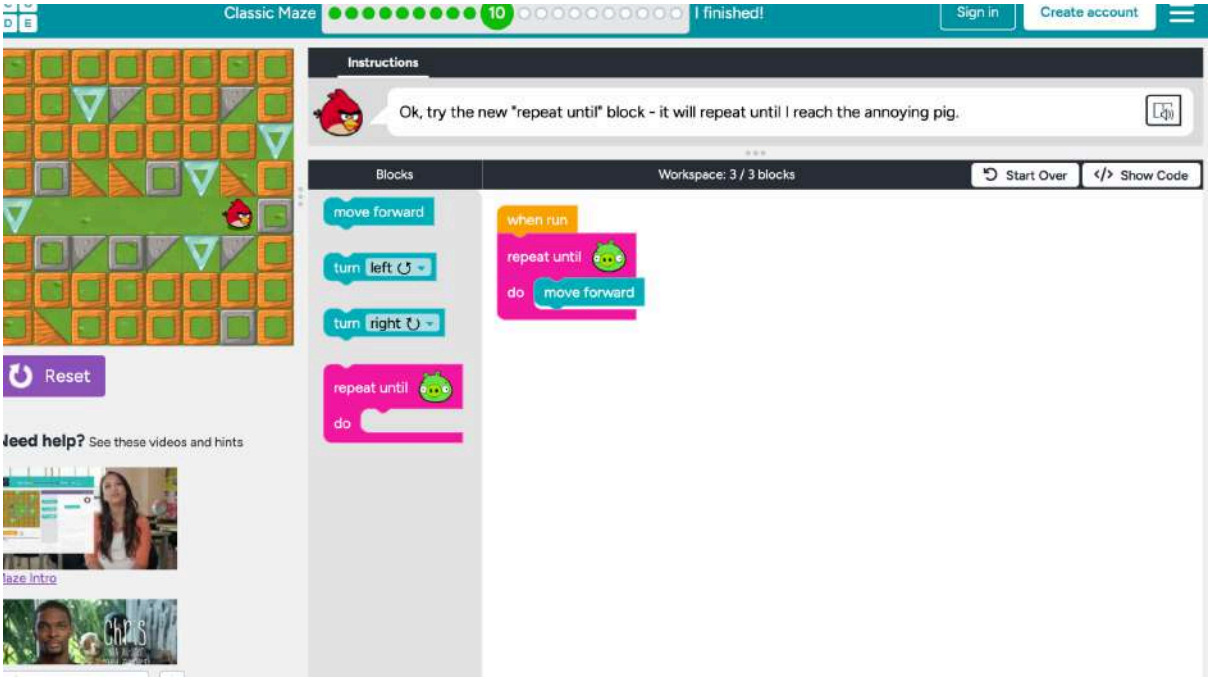
Dans cette étape, nous avons utilisé un bloc "répéter 3 fois".
Le personnage avance deux fois, puis tourne à droite, et cela se répète trois fois.

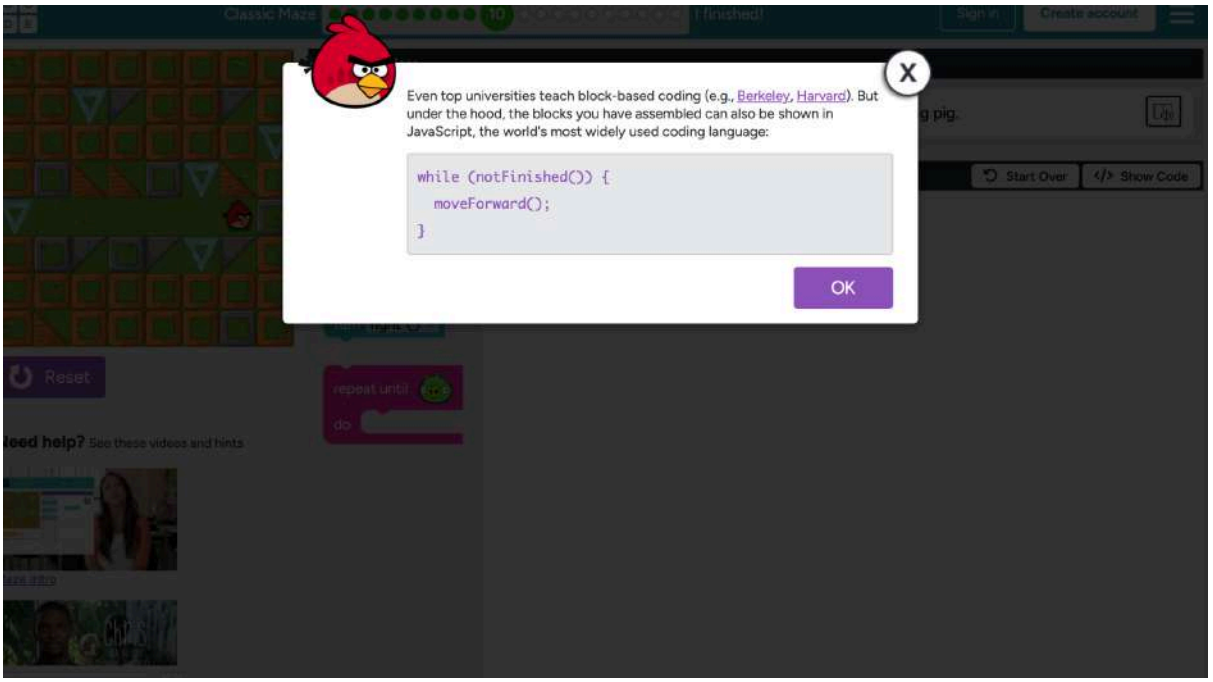




Etape 10

Dans cette étape, nous utiliserons une boucle while qui signifie : Tant que ce n'est pas fini, continue d'avancer. while (notFinished()) { → le personnage continue tant qu'il n'a pas atteint la fin du niveau.
moveForward(); → à chaque tour de boucle, il avance d'un pas.

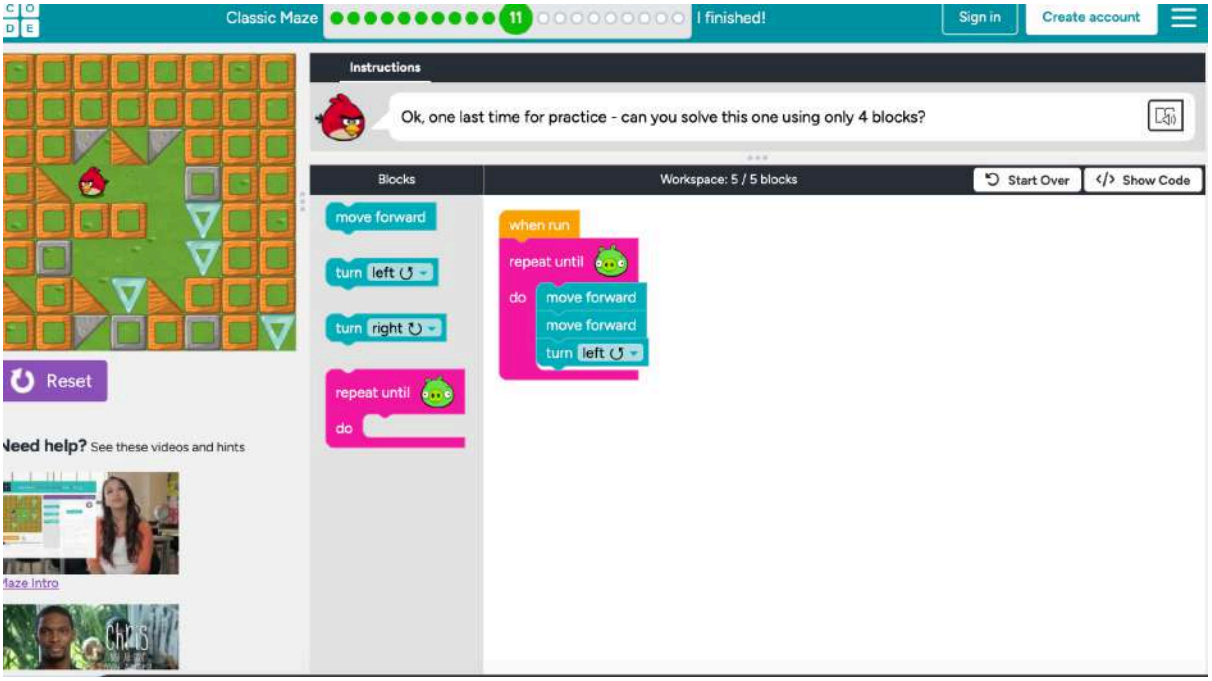


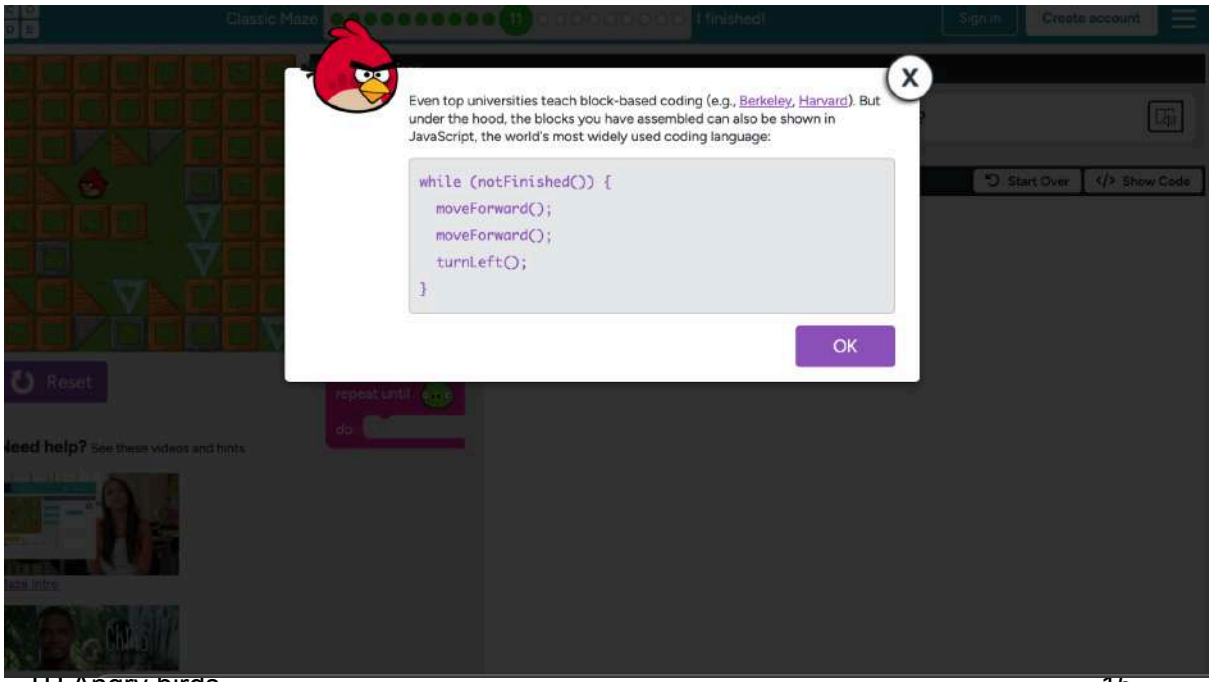


Etape 11

Ici nous avons utilisé une boucle while pour faire avancer le personnage deux fois puis tourner à gauche.

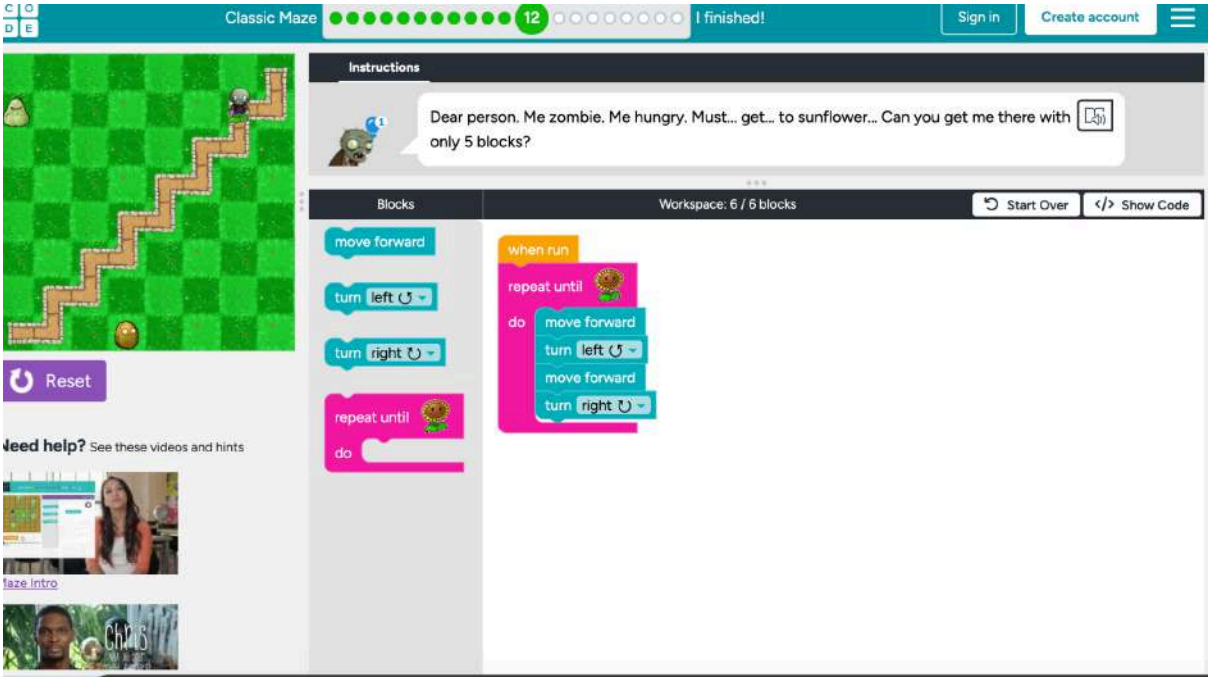
Cette séquence s’est répétée 2 fois.

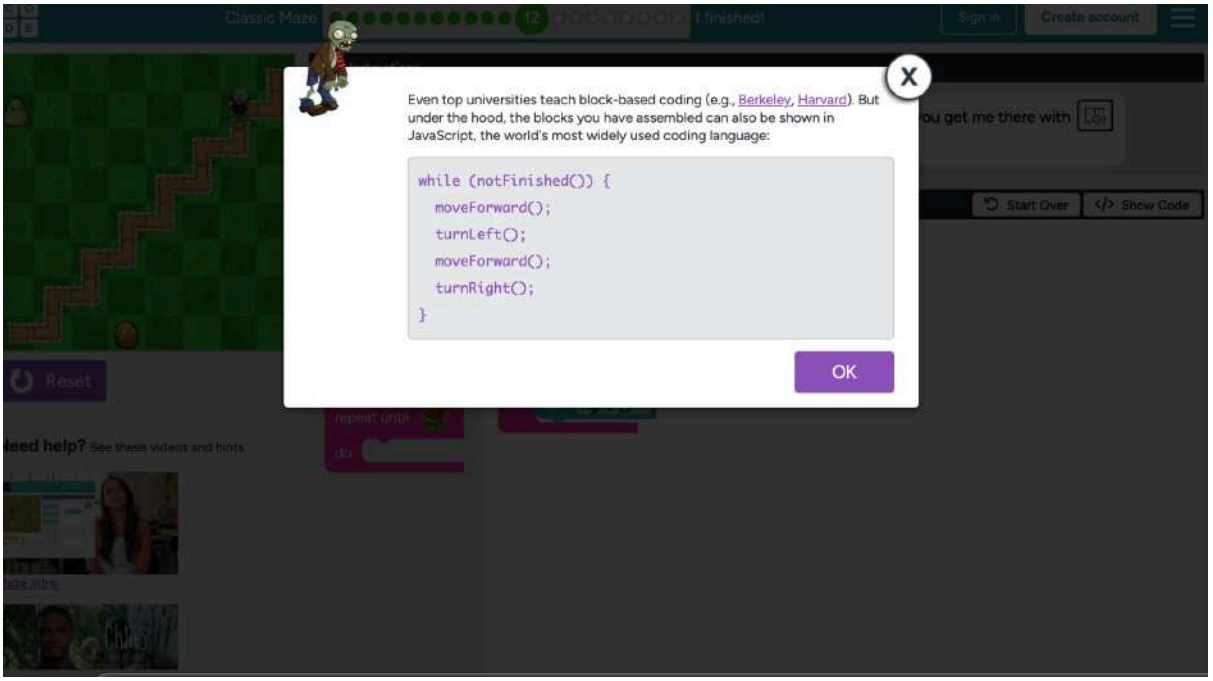




Etape 12

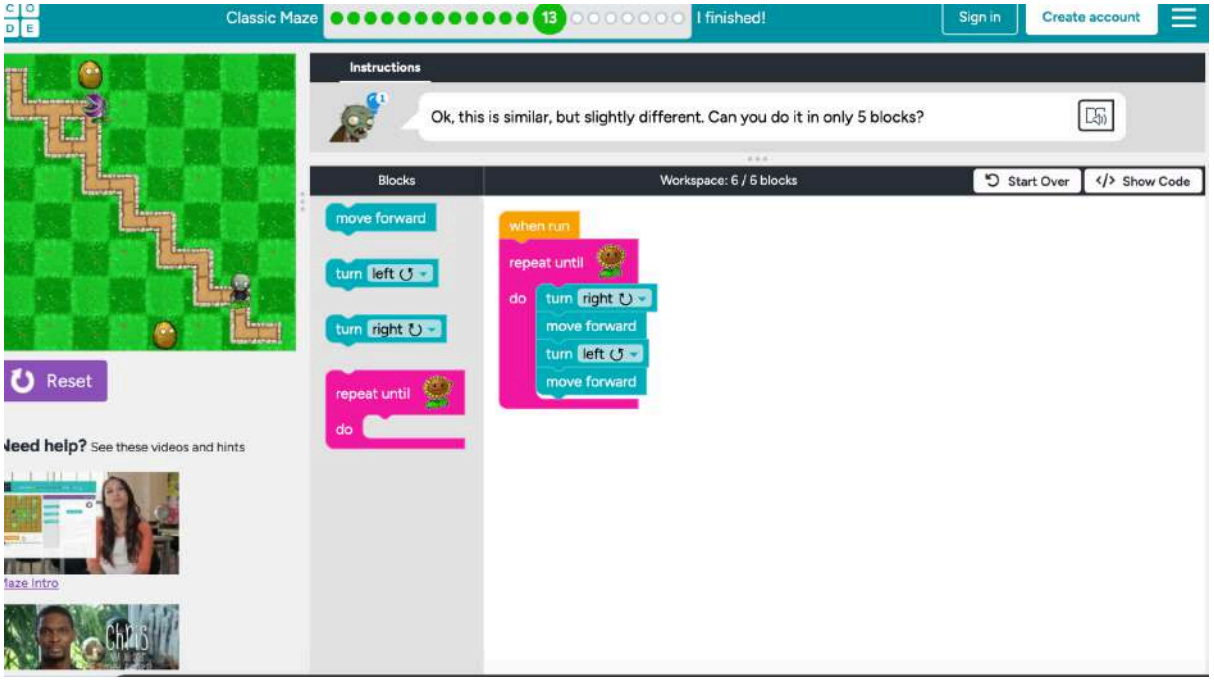
A présent le zombie tourne à gauche et à droite en permanence jusqu'à trouver la fleur avec l'instruction "tant que ce n'est pas fini".

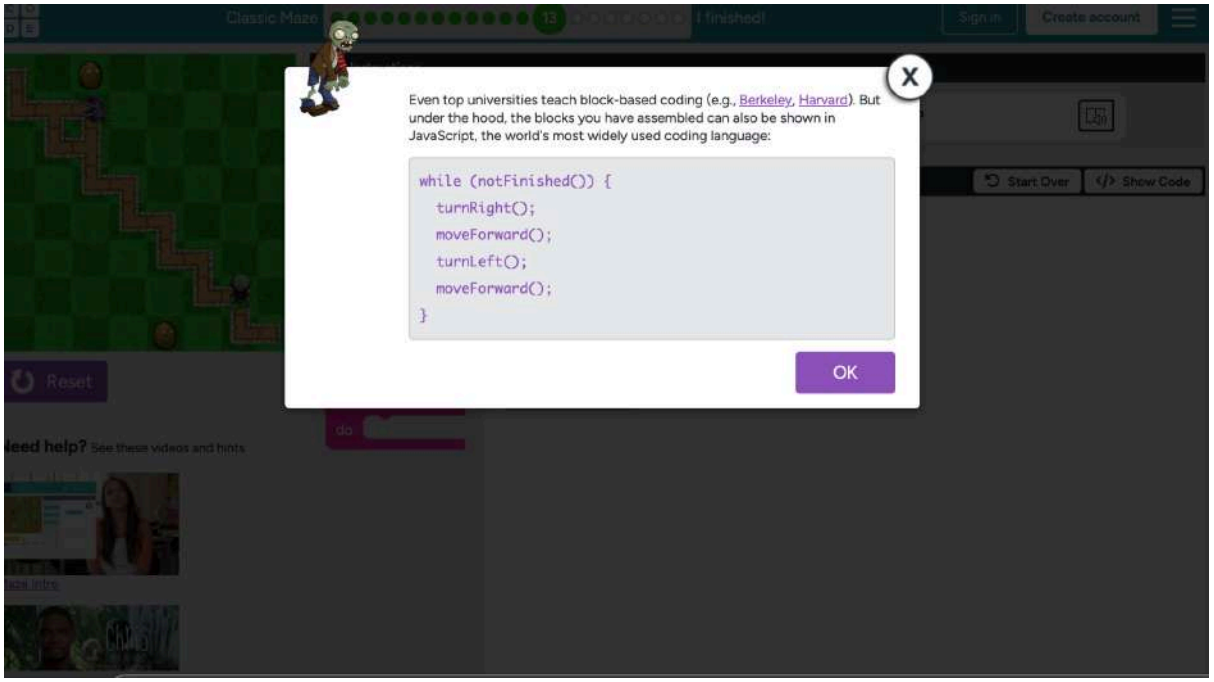




Etape 13

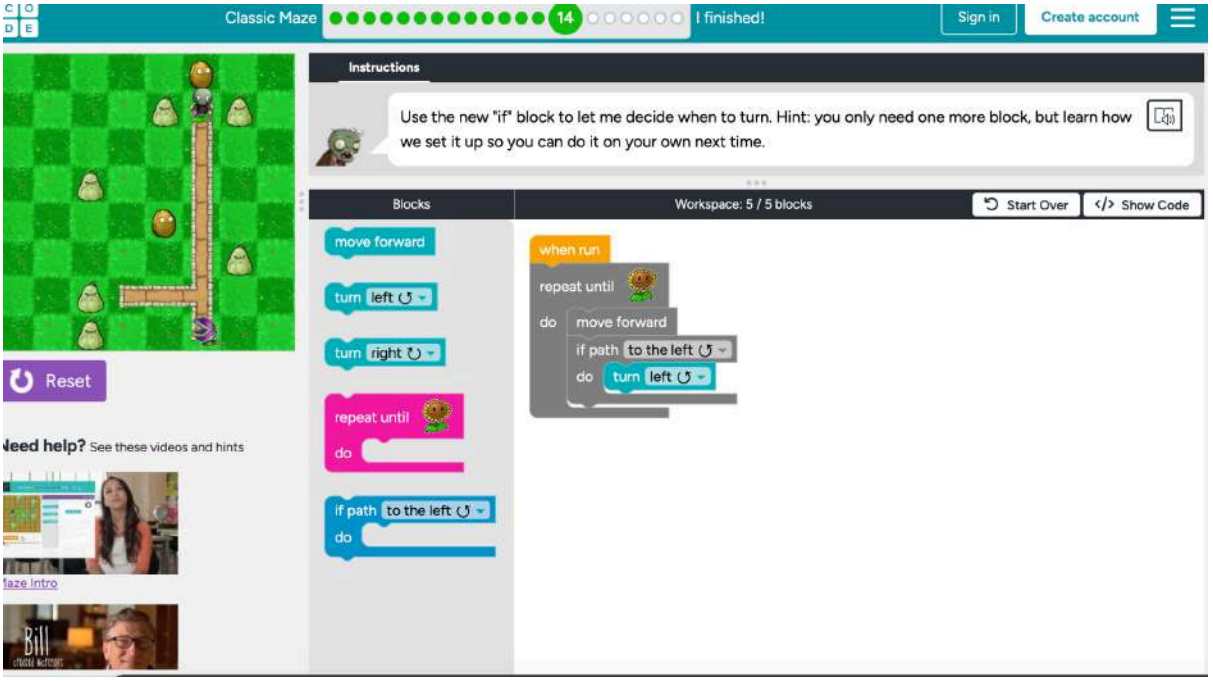
Nous faisons la même chose mais dans l’autre sens.

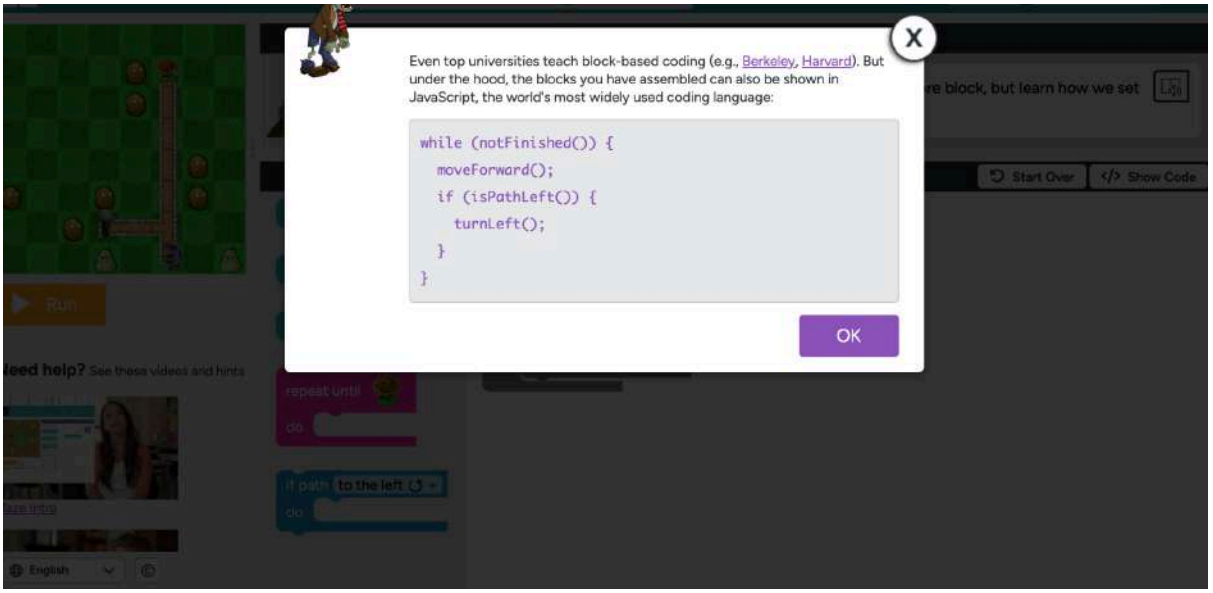




Etape 14

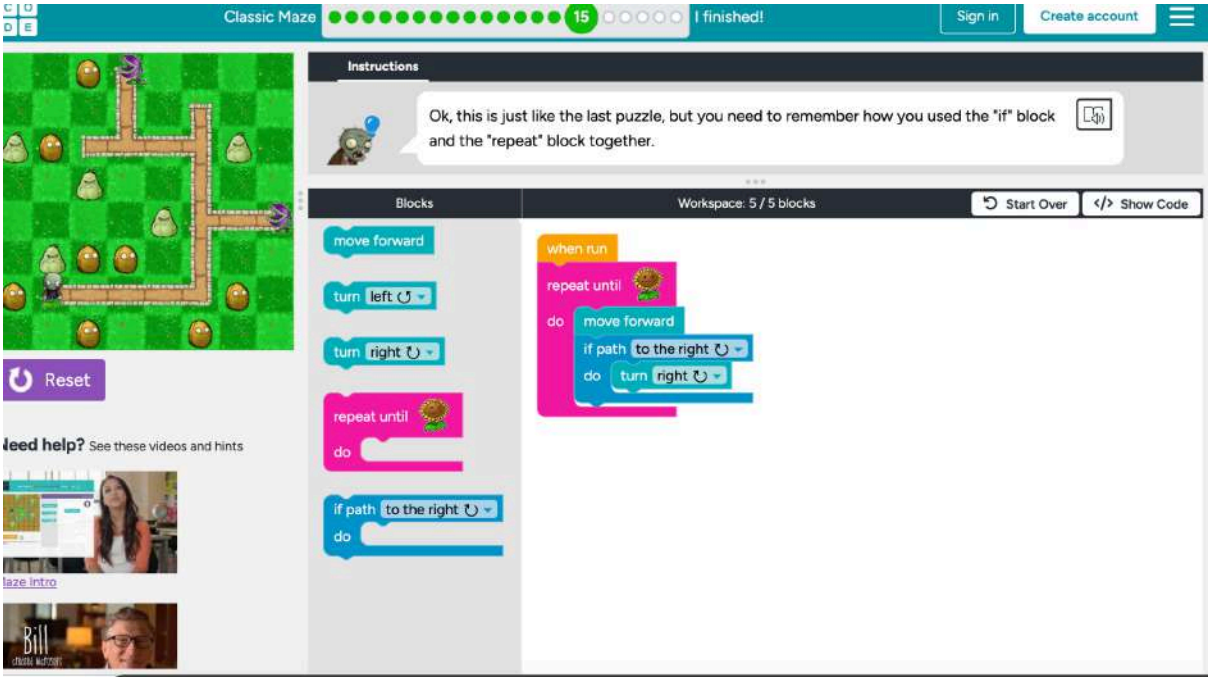
Dans cette étape, le zombie regarde s'il peut tourner à gauche, si c'est le cas il tournera donc à gauche.

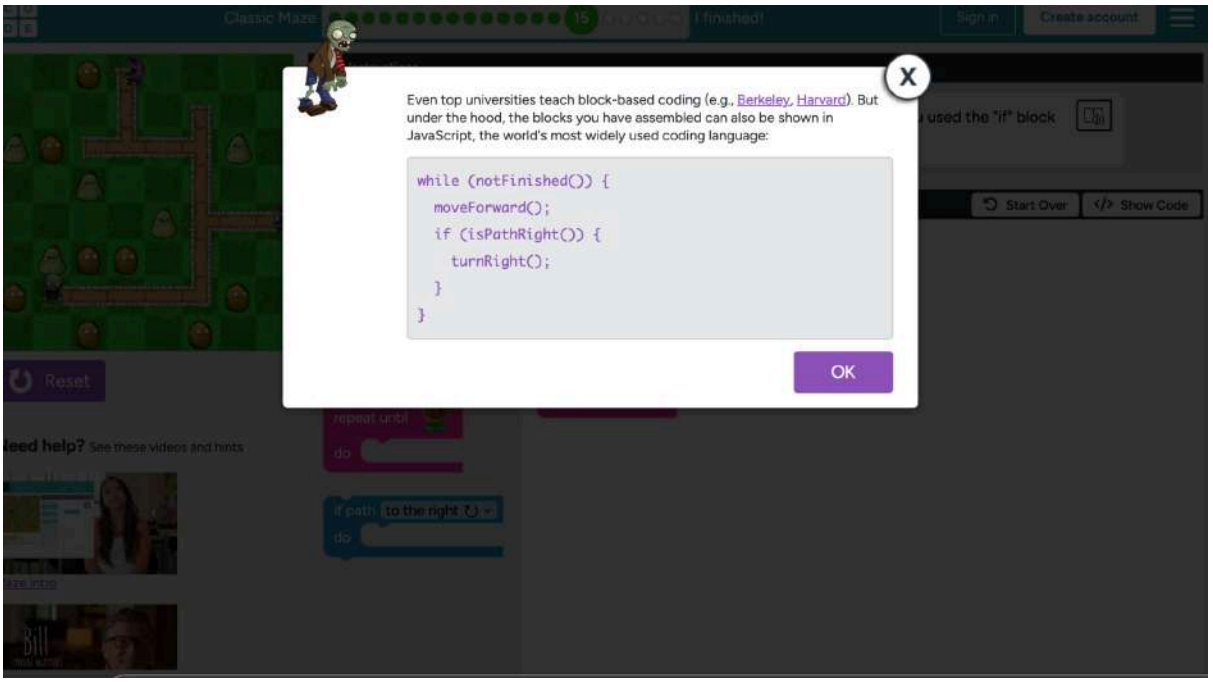




Etape 15

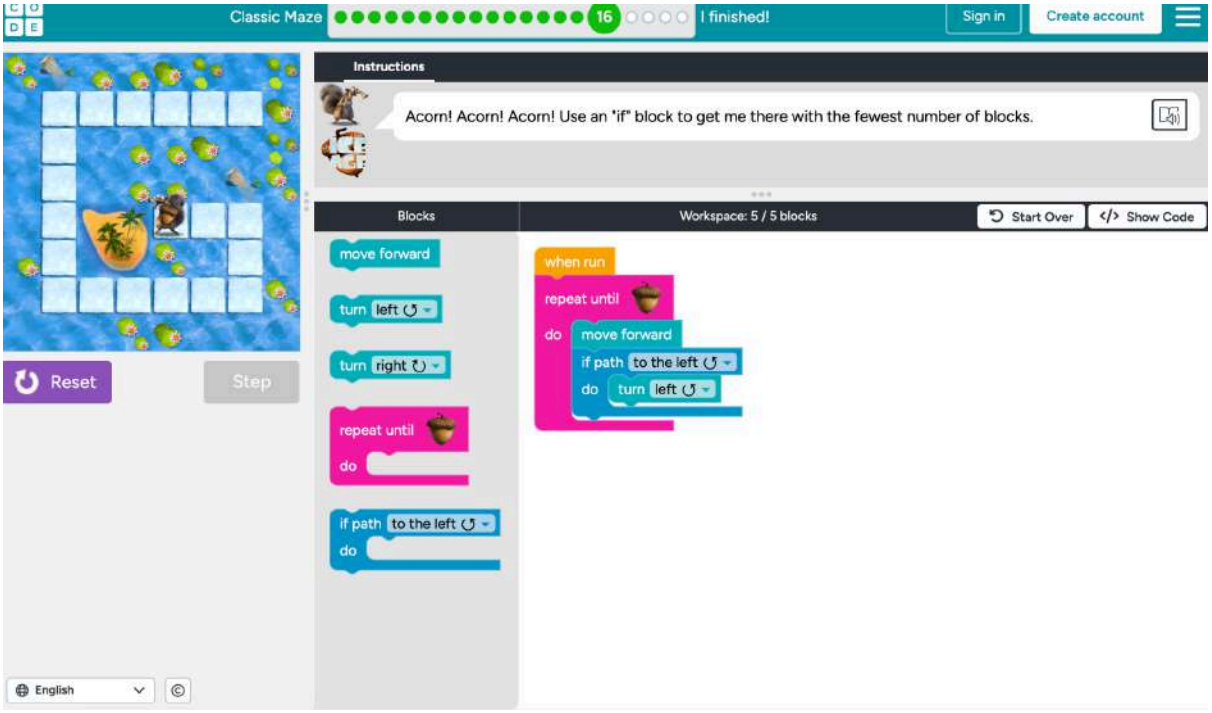
Ce code utilise une boucle while pour avancer tant que le parcours n’est pas fini.
À chaque pas, il vérifie s’il y a un chemin à droite et tourne si c’est possible, sinon il continue tout droit.

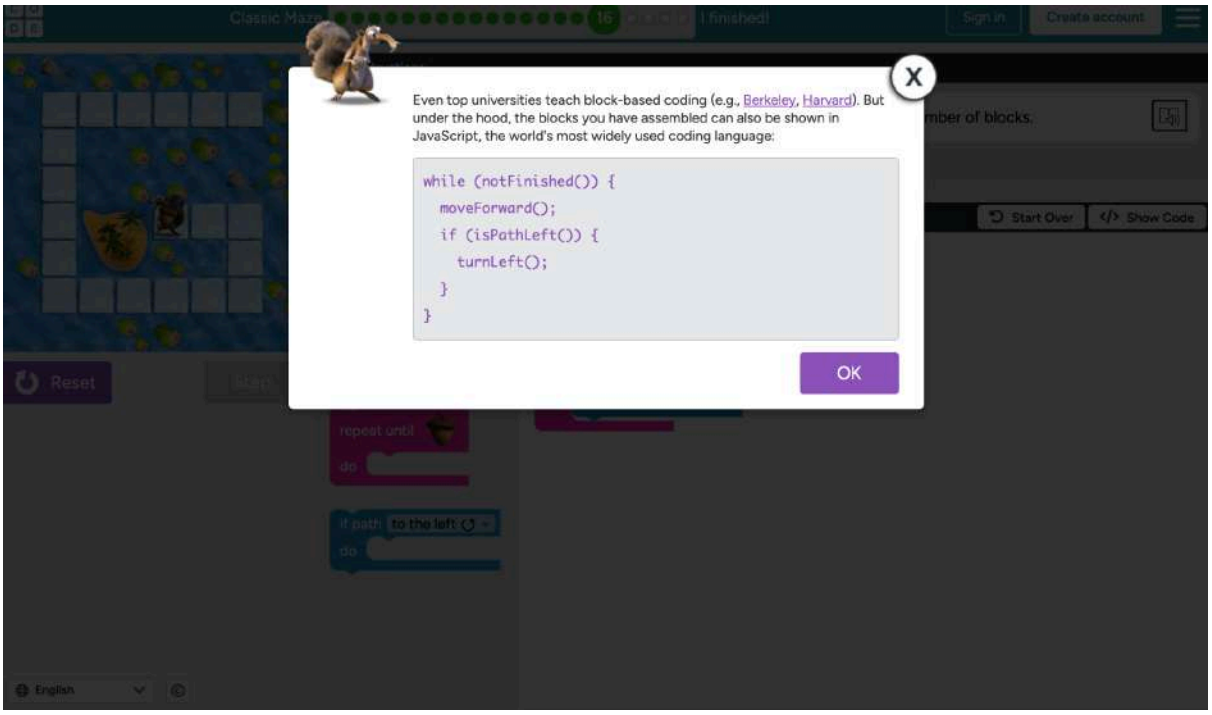




Etape 16

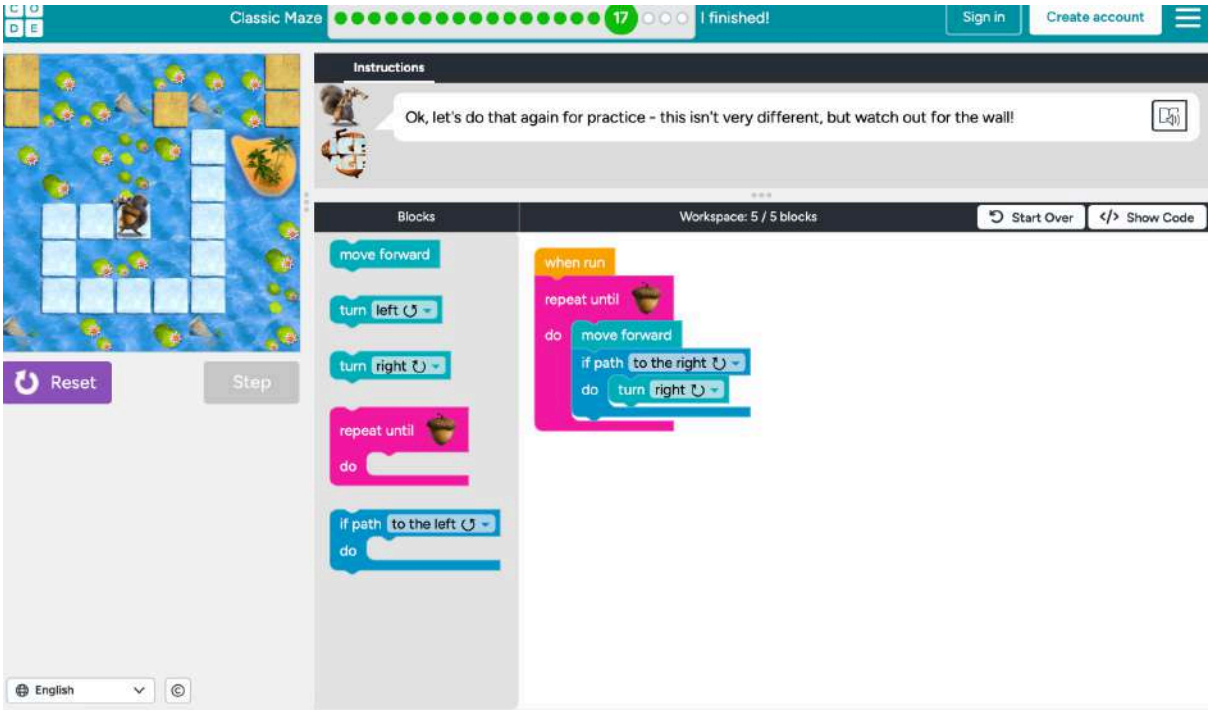
L’étape 16 est similaire à l’étape 15.

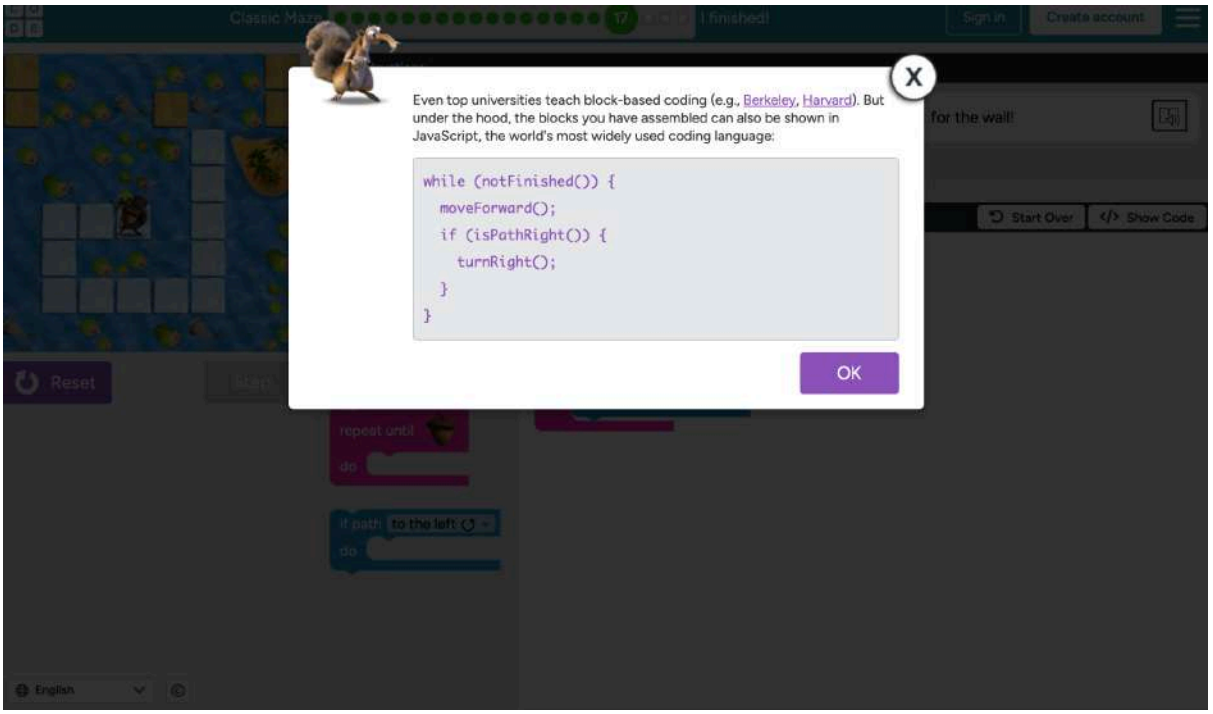




Etape 17

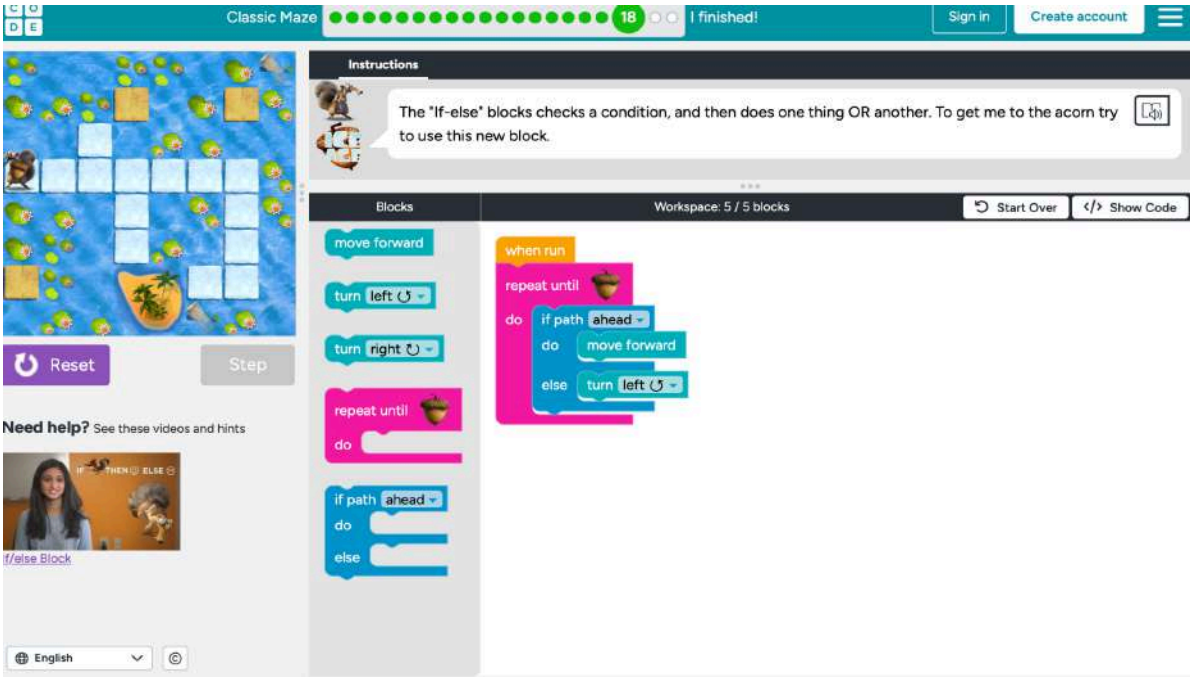
Nous faisons la même chose que pour les étapes précédentes.

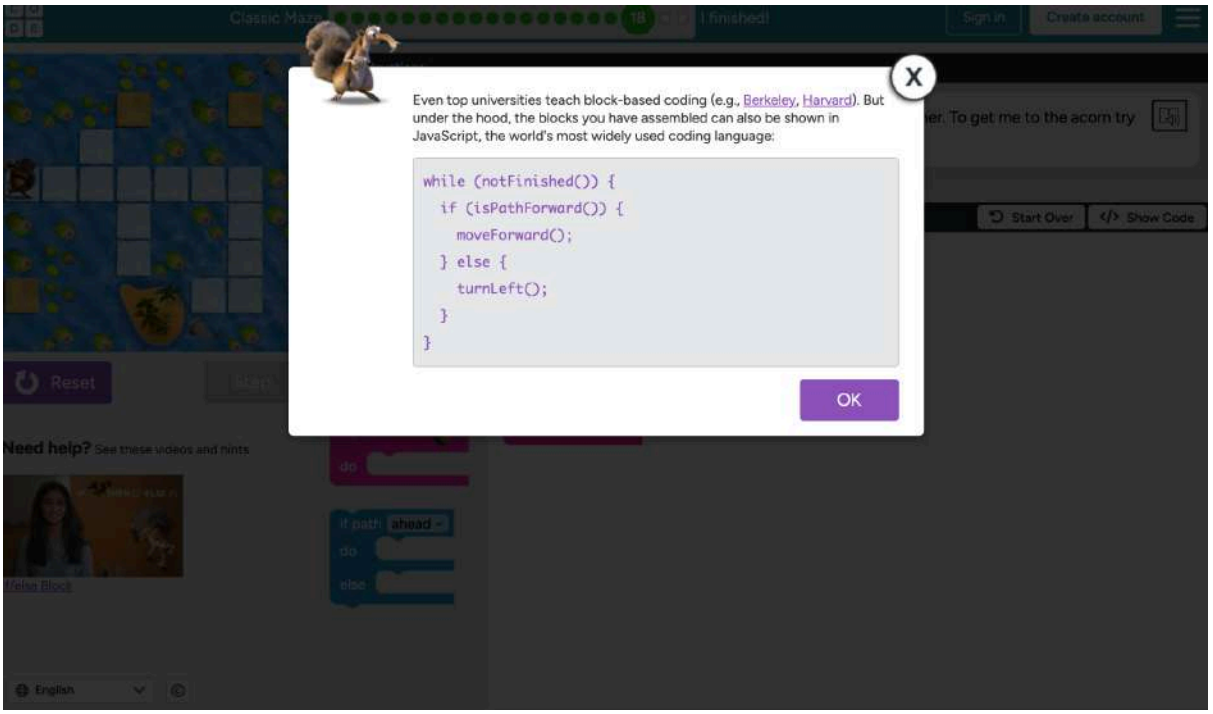




Etape 18

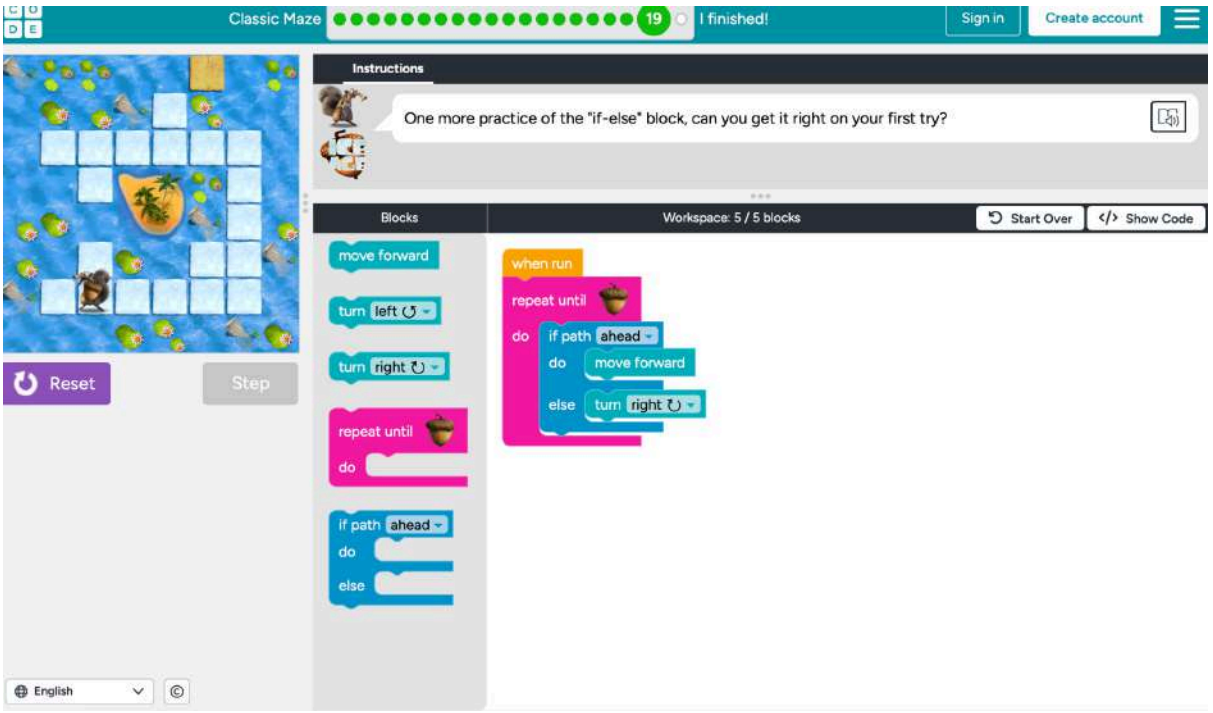
Expliquons pour bien comprendre ce que le code veut dire
Si la condition `if (isPathForward())` est vraie, l'écureuil avance d'un pas.
} else { → Sinon si il n'y pas de chemin devant on passe à l'option turnLeft(); et on tourne à gauche.

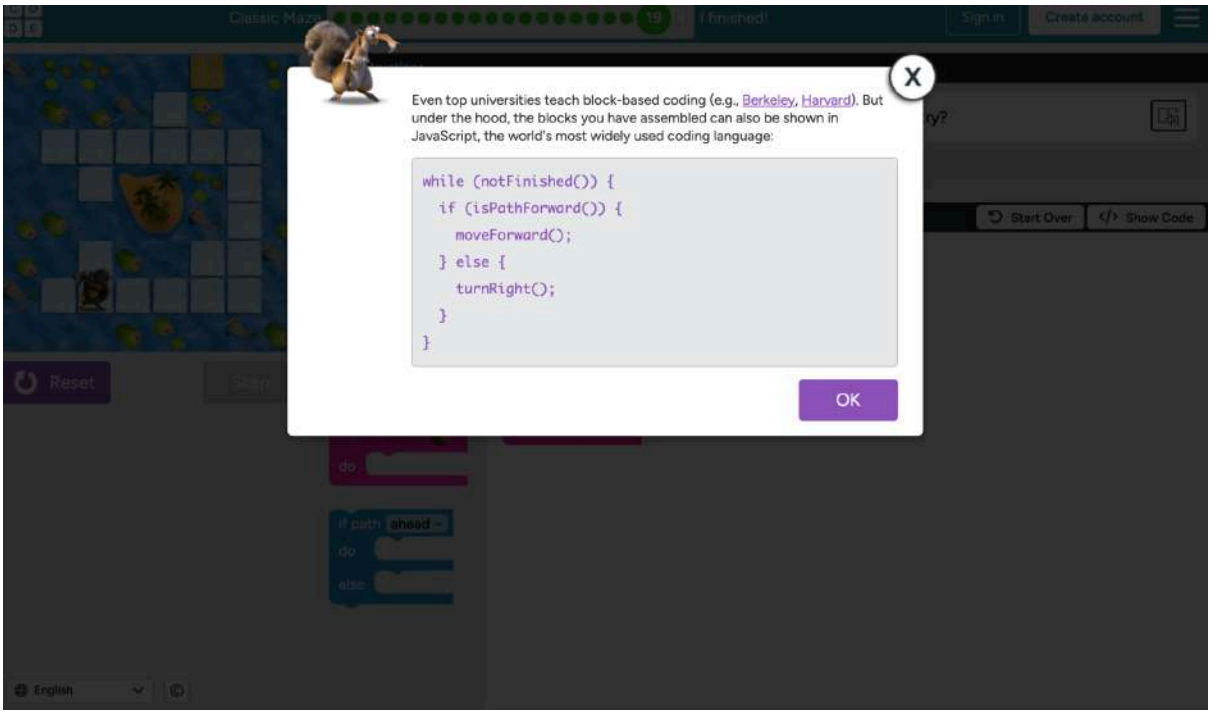




Etape 19

Même code que l'étape 18, à la différence que l'on ajoute turnRight(); pour tourner à droite.

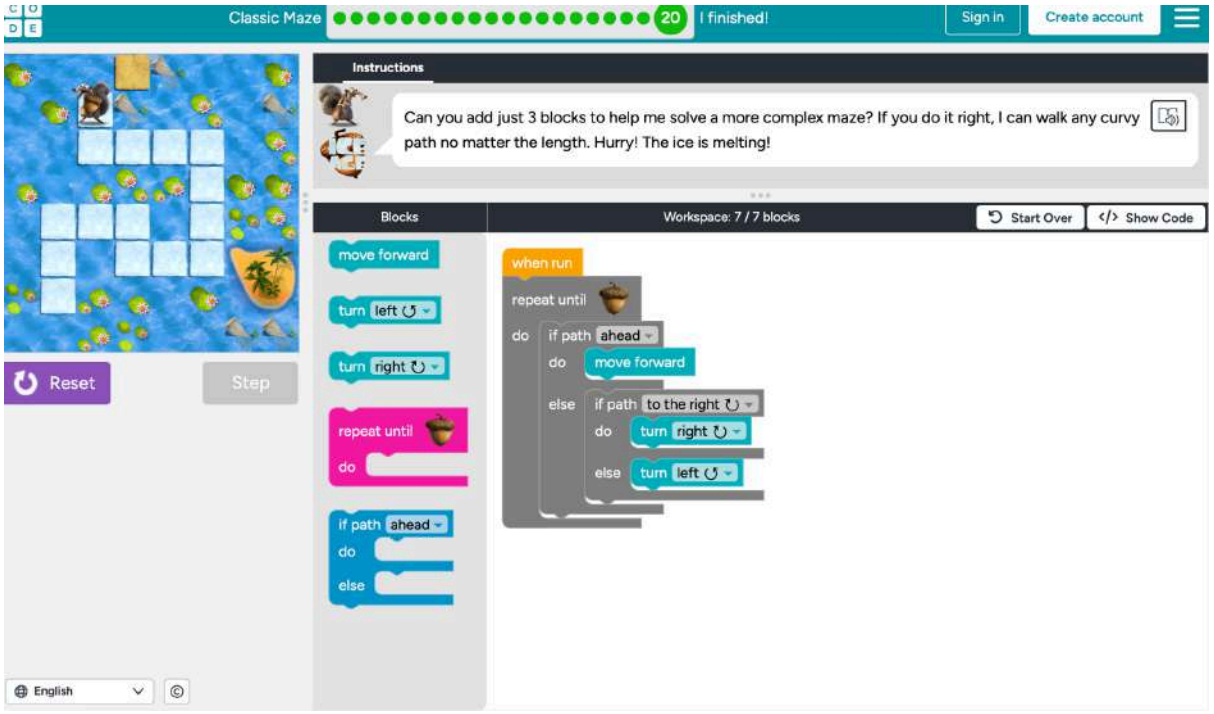




Etape 20

On termine ce TP avec l'étape 20 et une explication du code.

La boucle while (notFinished()) veut dire tant que le personnage n'a pas terminé le niveau, il continue d'exécuter les actions à l'intérieur.
if (isPathForward())
-> Si le chemin est libre devant, il avance (moveForward();).
else {
-> Si le chemin est bloqué devant, alors on vérifie une autre possibilité :
if (isPathRight())
-> Si le chemin est libre à droite, il tourne à droite (turnRight();).
else { turnLeft(); }
-> Sinon, s'il n'y a pas de chemin devant ni à droite, alors il tourne à gauche.



Fin.

Etape 20 suite

Visualisation du code.