The Maze Game	Mini Project 5: Create a specific maze

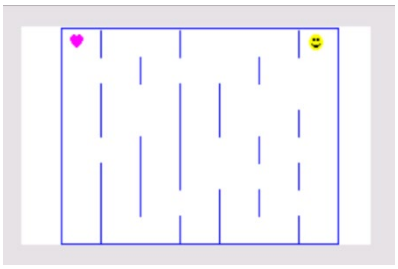
In this fifth mini-project, you will create a maze. You'll use lists to store values for each column indicating open and closed doors. You'll use **Pxl-On** to turn on pixels.

Objectives:

- Use Lists to store information
- Use If statements to make decisions
- Use Pxl-On to draw walls

The MAZE Game Project Overview:

After completing a series of 7 mini-projects, you will have a maze game similar to the one on the right. Projects 1 and 2 will provide skills needed to code movement in the maze game. Projects 3-6 will create code you'll import and use into your final project.



Mini-Project Order:

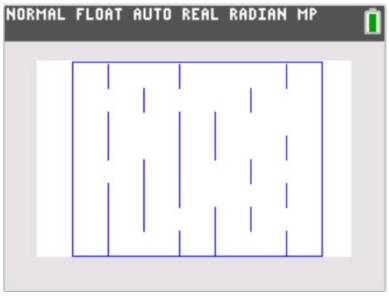
1. Detect which keys are pressed
2. Use key presses to move string
3. Draw objects using pixels
4. Move objects using keypresses and variables
5. **Create a specific Maze**
6. Randomize maze attributes
7. Create the final maze project.

Teacher Tip:

This file will be imported to project 6 make sure students give it the appropriate name so it can be found for project 6.

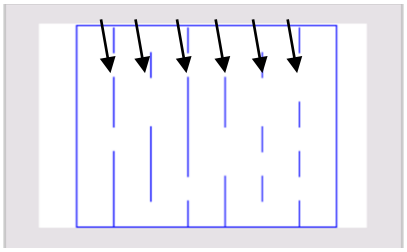
1. Create a new program named Maze.

   In this project you will use lists, loops and the pxl-on command to draw a maze similar to the one below.

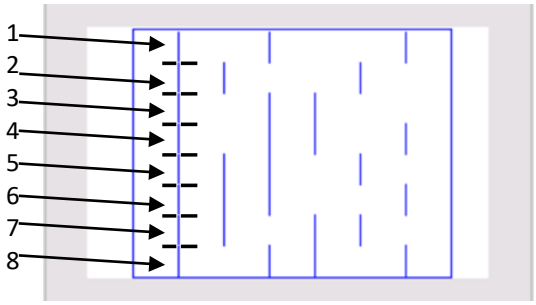


2. The maze has 6 columns of walls not including the boarder. (L1, L2…)
   The maze has 8 rows.
   You will use Lists to store the properties of the walls.

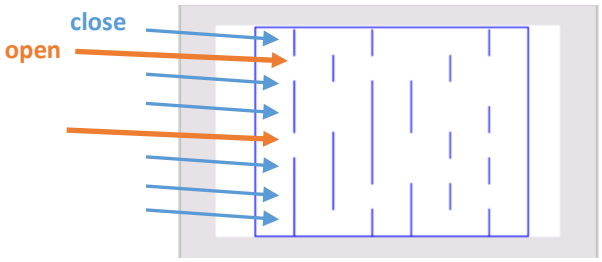
L1 L2 L3 L4 L5 L6



Each wall has 8 segments.

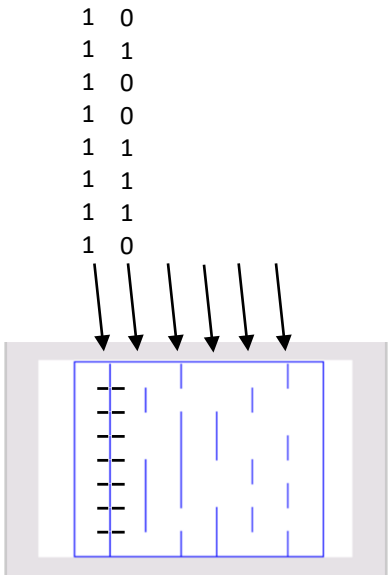


The segments are either **open** or **closed**.

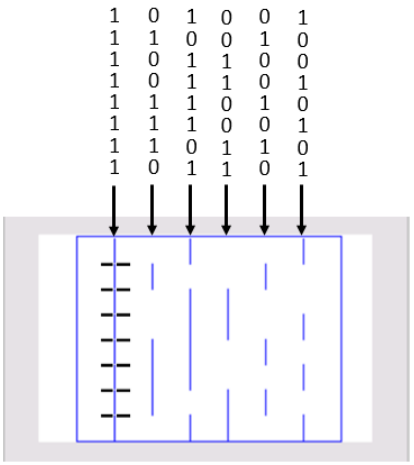


3. In each column, we will use the number 1 to present a "closed" door and a 0 to represent an "open" door. Each column will have eight values.

   Finish labeling the other 4 walls.



4. Does your picture match the one below?



You could also write the values to the left of the wall.

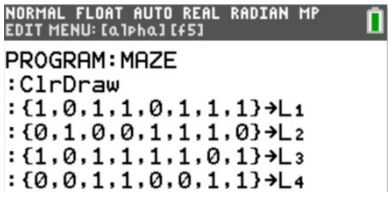
The picture below models the same maze on the left, only the first column has two open doors.

1	0	1	0	0	1	
0	1	0	0	1	0	
1	0	1	1	0	0	
1	0	1	1	0	1	
0	1	1	0	1	0	
1	1	1	0	0	1	
1	1	0	1	1	0	
1	0	1	1	0	1	

5. Now you need to code the status of the six columns.
   Store each list of numbers in a separate list.
   Make sure to use list {} notation to define your list.

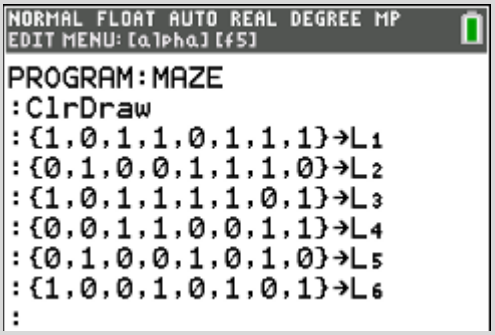
```
:ClrDraw
:{1,0,1,1,0,1,1,1} →L1
:{0,1,0,0,1,1,1,0} →L2
:{1,0,1,1,1,1,0,1} →L3
…Add in the code for L4, L5 and L6
```

1	0	1	0	0	1	
0	1	0	0	1	0	
1	0	1	1	0	0	
1	0	1	1	0	1	
0	1	1	0	1	0	
1	1	1	0	0	1	
1	1	0	1	1	0	
1	0	1	1	0	1	

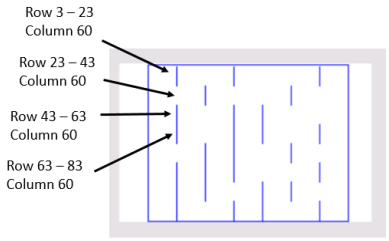


Teacher Tip:

A common mistake is to use parentheses instead of curly brackets. The lists don't have to match exactly what is shown. They need to have 8 items in each, all 1s and 0s. There has to be at least one 0 in each column to create a doorway.



6. Now you need to draw the walls using pixels.
Each wall segment has a length of 20 pixels.



The top wall has a 1 stored in L1(1).
To code the first door we could use the following.

(3 + 0, 60)
(3 + 1, 60)
(3 + 2, 60)
…
(3 + 20, 60)

The next wall is open because 0 is stored in L1(2).

The third wall is solid because 1 is stored in L1(3).

(3 + 20*2 + 0, 120)
(3 + 20*2 + 1, 120)
(3 + 20*2 + 2, 120)
…
(3 + 20*2 + 20, 120)

The fourth wall is solid because 1 is stored in L1(4).

(3 + 20*3 + 0, 150)
(3 + 20*3 + 1, 150)
(3 + 20*3 + 2, 150)
…
(3 + 20*3 + 20, 150)

Can you write a loop to draw the walls represented in L1?

7. L1(1)

(3 + 20*0 + 0, 60)
(3 + 20*0 + 1, 60)
(3 + 20*0 + 2, 60)
…
(3 + 20*0 + 20, 60)

L1(2)

Don't draw because it's holding a 0. Therefore, use an IF to only draw if there is a 1 in the list.

(3 + 20*1 + 0, 90)
(3 + 20*1 + 1, 90)
(3 + 20*1 + 2, 90)
…
(3 + 20*1 + 20, 90)

L1(3)

(3 + 20*2 + 0, 120)
(3 + 20*2 + 1, 120)
(3 + 20*2 + 2, 120)
…
(3 + 20*2 + 20, 120)

L1(4)

(3 + 20*3 + 0, 150)
(3 + 20*3 + 1, 150)
(3 + 20*3 + 2, 150)
…
(3 + 20*3 + 20, 150)

```
:For(A, 1, 8)
:If L1(A) = 1
:Then
:For(I,0,20)
:Pxl-On(3 + 20*(A-1) + I, 60)
:End
:End
```

Repeat the code for each List.

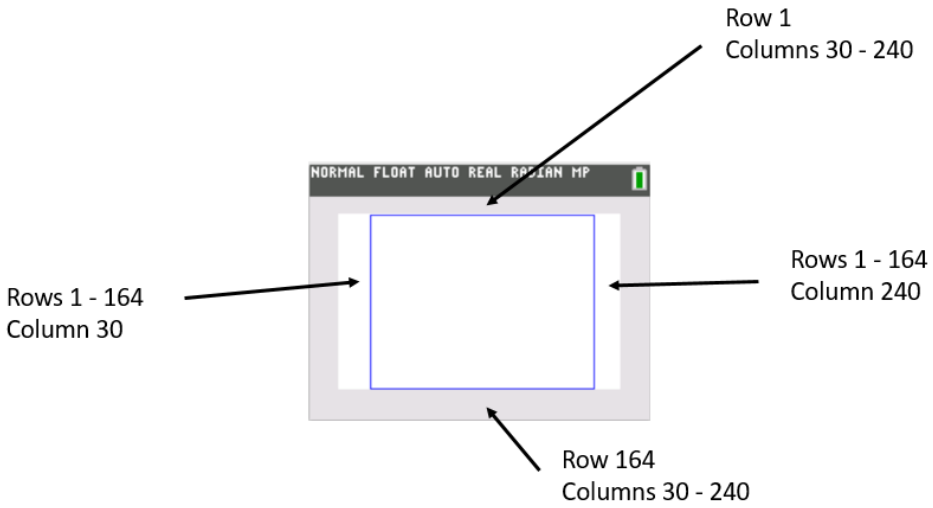
L2 should be 30 pixels to the right of L1. The pixel code would be

:Pxl-On(3 + 20*(A-1) + I, 90)

L3 should be 30 pixels to the right of L2. The pixel code would be

:Pxl-On(3 + 20*(A-1) + I, 120)

8. Code the four loops to create the four line segments for the border.



9. Does your code look something similar to the code on the right?

```
PROGRAM:MAZE
:
:For(A,30,240)
:Pxl-On(1,A,BLUE)
:Pxl-On(164,A)
:End
:For(A,1,164)
:Pxl-On(A,30)
:Pxl-On(A,240)
:End
```

Teacher Tip:
Students can skip project #6 if they don't want to have a random maze show each time the game is played.

```
PROGRAM:MAZE
:ClrDraw
:{1,0,1,1,0,1,1,1}→L1
:{0,1,0,0,1,1,1,0}→L2
:{1,0,1,1,1,1,0,1}→L3
:{0,0,1,1,0,0,1,1}→L4
:{0,1,0,0,1,0,1,0}→L5
:{1,0,0,1,0,1,0,1}→L6
:
:For(A,1,8)
:
:If L1(A)=1
:Then
:For(I,0,20)
:Pxl-On(3+20*(A-1)+I,30*2)

:End
:End
:
```

```
:If L₂(A)=1
:Then
:For(I,0,20)
:Pxl-On(3+20*(A-1)+I,30*3)

:End
:End
:
:If L₃(A)=1
:Then
:For(I,0,20)
:Pxl-On(3+20*(A-1)+I,30*4)

:End
:End
:
:If L₄(A)=1
:Then
:For(I,0,20)
:Pxl-On(3+20*(A-1)+I,30*5)

:End
:End
:
:If L₅(A)=1
:Then
:For(I,0,20)
:Pxl-On(3+20*(A-1)+I,30*6)

:End
:End
:
:If L₆(A)=1
:Then
:For(I,0,20)
:Pxl-On(3+20*(A-1)+I,30*7)
:End
:End
:
:End
:
:
:For(A,30,240)
:Pxl-On(1,A,BLUE)
:Pxl-On(164,A)
:End
:For(A,1,164)
:Pxl-On(A,30)
:Pxl-On(A,240)
:End
```