The Maze Game — Mini Project 7: Final Maze Project

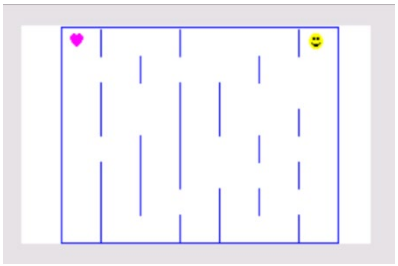
In the final project, you will now create the Maze Game. You'll import the *player*, *goal* and *random* maze files. You'll add code to restrict movement to the maze. Finally, you'll create a message that lets the user know when the goal is obtained.

Objectives:

- Use If statements to make selections
- Use a While loops to repeat code
- Use For loops to repeat code
- Use Pxl-On and Pxl-off to draw pixels
- Use Lists to store values

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.

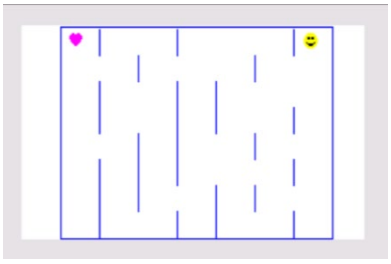


Mini-Project Order:

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.**

Tech Tip:
You will import the projects from 3, 4 and 6 to finish this project. If you didn't do Mini-project 6, you can import 5 in its place, but the maze will not be random if this substitution is made.

1. You now have most of the components needed to create the maze game. You will import code from other projects and make some small alterations and additions.



File to Import	Modifications
Player (Project 4)	Only move left/right if a door
RMAZE (Project 6)	
GOAL (Project 3) (Sample is SMILE)	Modify the location if needed

2. A.) Create a final project MZGAME

   B.) Add a comment line, then import your maze created in project 6 named RMAZE using the recall method.

   ```
   :"MAZE
   :rcl RMAZE
   ```

   C.) Add a comment then import your goal object created in project 3 using the recall method. The sample was names SMILE, your file might be different.

   ```
   :"GOAL
   :rcl SMILE
   ```

   Remove the ClrDraw from the beginning of the goal code

   D.) Add a comment line then import your player code.

   ```
   :"PLAYER CODE
   :rcl PLAYER
   ```

   Remove the ClrDraw from the beginning of the Player code

   E.) Run your code and make sure you don't have any errors.
   If all the objects don't stay on the screen, go line by line through your code looking for extra ClrDraw statements.

   You should have a ClrDraw at the beginning.

3. Can you modify your code object to move 20 pixels when you move up vertically only IF it keeps your object on in the maze?

   Can you modify your object 20 pixels when you move down vertically IF it keeps your object on in the maze?

4. Did your modifications to your player work? Does your player stay in the maze and only move if a wall isn't present?

   Here is one possible way to accomplish appropriate vertical movement.

Top Border

```
:If K=25 and V≥20
:Then
:V-20→V
:ClrDraw
:End
```

Bottom B

```
:If K=34 and V≤120
:Then
:V+20→V
:ClrDraw
:End
```

5. You need to modify the left and right arrow keys from the PLAYER code. You first need to know how many spaces down and over your player has traveled. Inside the while loop for your key press, write the following:

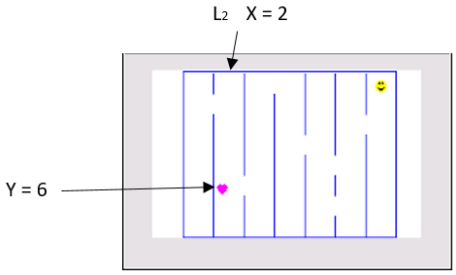
   :V/20 + 1 →Y

   :H/30 + 1 →X

```
:V/20+1→Y
:H/30+1→X
```

6. How do you restrict the player so it doesn't walk through walls?

   The Y value will let you know which row you are on. You will use this when determining if the element in list is a 0 (no wall) or a 1 (wall). The X value will let you know which List to analyze.



L2(6) = 0 The right arrow key should move the object right.

L1(6) = 1 The left arrow key should NOT move the object left.

In this particular example,

to move right you would check	to move left you would check
If X = 2 and L2(6) = 0	If X - 1 = 1 and L1(6) = 0
Move right	Move left

7. Locate your of statement to move left (If K = 24) and your move right (If K = 26).

   Here is one way to modify the code.

Left

Include L_3, L_4, L_5

```
:If K=24 and X > 1
:Then
:If X-1 = 1 AND L1(Y)=0
:Then
:H-30 →H
:End
:If X-1 = 2 AND L2(Y)=0
:Then
:H-30 →H
:End
.....
:End
```

Right

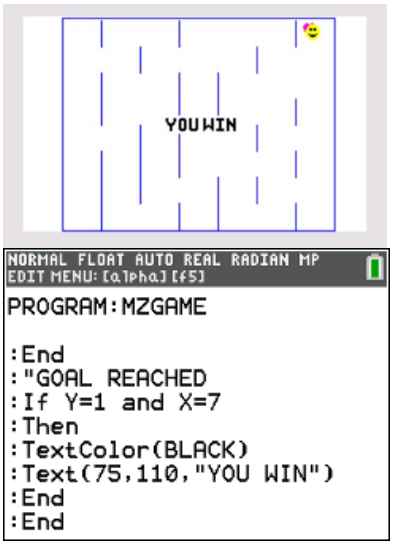
Include L_4, L_5, L_6

```
:If K=26 and X ≤ 6
:Then
:If X = 1 AND L1(Y)=0
:Then
: H+30 →H
:End
:If X = 2 AND L2(Y)=0
:Then
:H+30 →H
:End
.....
:End
```

CHECK your game. Make sure it works before you continue. Fix any errors you find.

8. How do you let the player know he or she has reached the goal?
   At the end your project above the last End statement that controls the key press loop, add the following lines

```
:"GOAL REACHED
:If Y=1 and X=7
:Then
:TextColor(BLACK)
:Text(75,110,"YOU WIN")
:End
```



9. Optional: Add sound using a TI-Innovator™ Hub

Connect your calculator to a TI-Innovator Hub.

Use the “Set Sound” command in each of the four If statements for the arrow keys. You could use the same tone or a different tone for each key. Don’t forget to connect your TI-Innovator hub for the sound to work.

```
:If K=24 and X > 1
:Send(“SET SOUND 262 Time 0.5”)
:If X-1 = 1 AND L1(Y)=0
:Then
:H-30 →H
:End
:If X-1 = 1 AND L2(Y)=0
:Then
:H-30 →H
:End
…….
:End
```