The Maze Game	Mini-Project 3: Draw objects using pixels

In this third mini-project, you will design two characters a player character and a goal character. You'll create these characters one pixel at a time.

Objectives:

- Use the Pxl-On command to color specific pixels
- Use the Pxl-Off command to turn off pixels
- Use For loops to simplify and repeat code
- Create a character using pixels

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. **<u>Draw objects using pixels</u>**
4. Move objects using keypresses and variables
5. Create a specific Maze
6. Randomize maze attributes
7. Create the final maze project.

Teacher Tip:
Graph paper and colored pencils are useful to create pictures.

1. Think of your calculator as a piece of graph paper.
   The graph paper has 164 rows and 264 columns.
   To color one pixel the code is:
   Pxl-On(row number, column number, color)

   To create a blue rectangle that has a width of five and a height of two type the code below.
   AxesOff is under FORMAT. (2nd zoom).
   Colors can be found using the [prgm] > COLORS.
   All other commands are under DRAW. (2nd prgm)

```
:AxesOff
:BackgroundOff
:ClrDraw
:Pxl-On(0, 0, BLUE)
:Pxl-On(0, 1, BLUE)
:Pxl-On(0, 2, BLUE)
:Pxl-On(0, 3, BLUE)
```

©2020 Texas Instruments Incorporated

```
:Pxl-On(0, 4, BLUE)
:Pxl-On(1, 0, BLUE)
:Pxl-On(1, 1, BLUE)
:Pxl-On(1, 2, BLUE)
:Pxl-On(1, 3, BLUE)
:Pxl-On(1, 4, BLUE)
```

Teacher Tip:
Students often intermix Pt-On and Pxl-On. Pt-On draws points on a coordinate grid using current window settings. Pxl-On is independent of the window setting. When debugging make sure all lines use Pxl-On.

2. Modify your code.
   Create a brown rectangle that has a width of 20 and a height of 3.

   The first step had you write a line of code for each pixel.
   To make a 3 row x 20 column BROWN rectangle, you would need 60 lines of code!
   To simplify this process, we will use a loop.

```
:AxesOff
:BackgroundOff
:ClrDraw
:For(C,0,19)
:Pxl-On(0, C, BROWN)
:Pxl-On(1, C, BROWN)
:Pxl-On(2, C, BROWN)
:End
```

```
PROGRAM:RECTANGL
:AxesOff
:BackgroundOff
:ClrDraw
:For(C,0,19)
:Pxl-On(0,C,BROWN)
:Pxl-On(1,C,BROWN)
:Pxl-On(2,C,BROWN)
:End
```

   *If you want to color a row, but turn a few points off, use the command
   Pxl-Off(row number, column number)

Teacher Tip:
The loop has uses [0,19] as the values for C. Unlike the Output command, the rows and columns for pixels start at 0. Students could experiment with different values for the loop.

More advanced users might use a nested For statement:

```
:For(C,0,19)
:For(R,0,2)
:Pxl-On(R, C, BROWN)
:End
:End
```

3. For the maze project you need to create two objects. Each object will be coded in its own program. One object will be your player that moves on the screen. The other will be a stationary goal object. Your objects need to be between 10 and 15 pixels in width and height.

   Use graph paper to sketch out each of your objects.

   Sample Objects:

Heart.8xp

Smile.8xp

4. Here's some sample code for the first sample object. If you designed your own objects, the snippets might give you help coding your own designs. Don't start coding your characters until you've read step 6.

```
:For(C,1,10)
:Pxl-On(3, C, MAGENTA)
:Pxl-On(4, C, MAGENTA)
:Pxl-On(5, C, MAGENTA)
:End
```

```
:For(C,3,8)
:Pxl-On(1, C, MAGENTA)
:End
:Pxl-Off(1, 5)
:Pxl-Off(1, 6)
```

```
:For(C,2,9)
:Pxl-On(2, C, MAGENTA)
:End
```

   Don't start coding your characters until you've read step 6.

Teacher Tip:
Make sure students create each object in its own file. They will both be imported into later projects.

5. Here's some sample code for the second sample object. If you designed your own objects, the snippets might give you help coding your own designs. Don't start coding your characters until you've read step 6.

```
:For(C,3,13)
:Pxl-On(6, C, YELLOW)
:Pxl-On(7, C, YELLOW)
:Pxl-On(8, C, YELLOW)
:Pxl-On(9, C, YELLOW)
:Pxl-On(10, C, YELLOW)
:End
:Pxl-Off(6,3)
:Pxl-Off(6,13)
:For(R,6,7)
:Pxl-On(R,6,Black)
:Pxl-On(R,7,Black)
:Pxl-On(R,9,Black)
:Pxl-On(R,10,Black)
:End
:Pxl-On(10,5,Black)
:Pxl-On(10,11,Black)
```

```
:For(C,6,10)
:Pxl-On(2, C,YELLOW)
:End
```

6. The player character will be placed in the upper left-hand corner of the maze. Code your PLAYER program so your character will be ready for import in project 7.

   The goal character will be placed in the upper right hand corner of the maze. Code your GOAL program so your character will be ready for import into project 7.

Goal object
Initial Row ≈ 10
Initial Column ≈ 220

Player object
Initial Row ≈ 12
Initial Column ≈ 35

*It might be useful to know how to copy and paste lines of code.
Press [alpha] [graph] for the f5 menu that contains copy and paste options.

Teacher Tip:

It is easier to have students adjust the alignment for the player and goal now while they are still in individual files.

Sample Code:

Heart

```
NORMAL FLOAT AUTO REAL DEGREE MP
EDIT MENU: [alpha] [f5]
PROGRAM:HEART
:ClrDraw
:AxesOff
:
:For(A,33,38)
:Pxl-On(12,A,MAGENTA)
:End
:Pxl-Off(12,35)
:Pxl-Off(12,36)
:
:For(A,32,39)
:Pxl-On(13,A,MAGENTA)
:End
:
:For(R,14,16)
:For(C,31,40)
:Pxl-On(R,C,MAGENTA)
:End
:End
:For(R,17,18)
:For(C,32,39)
:Pxl-On(R,C,MAGENTA)
:End
:End
:
:For(C,33,38)
:Pxl-On(19,C,MAGENTA)
:End
:
:For(C,34,37)
:Pxl-On(20,C,MAGENTA)
:End
:
:Pxl-On(21,35,MAGENTA)
:Pxl-On(21,36,MAGENTA)
:
:
```

Smile

```
NORMAL FLOAT AUTO REAL DEGREE MP
EDIT MENU: [alpha] [f5]
PROGRAM:SMILE
:CoordOff
:ClrDraw
:
:Pxl-On(10,223,YELLOW)
:Pxl-On(10,224,YELLOW)
:Pxl-On(10,225,YELLOW)
:Pxl-On(10,226,YELLOW)
:Pxl-On(10,227,YELLOW)
:
:For(C,222,228)
:Pxl-On(11,C,YELLOW)
:End
:
:For(C,221,229)
:Pxl-On(12,C,YELLOW)
:Pxl-On(13,C,YELLOW)
:End
:
:For(R,12,13)
:Pxl-On(R,223,BLACK)
:Pxl-On(R,224,BLACK)
:Pxl-On(R,226,BLACK)
:Pxl-On(R,227,BLACK)
:End
:
:For(C,220,230)
:Pxl-On(14,C,YELLOW)
:Pxl-On(15,C,YELLOW)
:Pxl-On(16,C,YELLOW)
:Pxl-On(17,C,YELLOW)
:End
:
:Pxl-On(17,228,BLACK)
:Pxl-On(17,222,BLACK)
:
:Pxl-On(18,221,YELLOW)
:Pxl-On(18,222,YELLOW)
:Pxl-On(18,223,BLACK)
:Pxl-On(18,224,BLACK)
:Pxl-On(18,225,BLACK)
:Pxl-On(18,226,BLACK)
:Pxl-On(18,227,BLACK)
:Pxl-On(18,228,YELLOW)
:Pxl-On(18,229,YELLOW)
:
```

```
:Pxl-On(19,221,YELLOW)
:Pxl-On(19,222,YELLOW)
:Pxl-On(19,223,YELLOW)
:Pxl-On(19,224,BLACK)
:Pxl-On(19,225,BLACK)
:Pxl-On(19,226,BLACK)
:Pxl-On(19,227,YELLOW)
:Pxl-On(19,228,YELLOW)
:Pxl-On(19,229,YELLOW)
```

```
:
:For(C,224,228)
:Pxl-On(20,C,YELLOW)
:End
:
:For(C,223,227)
:Pxl-On(21,C,YELLOW)
:End
```