FOREVER MAZE GAME
GROUP 20

CHARLES RAGONA, JENNIFER BUSS, MAXWELL DANKU, SAYONTAN ROY

911: WHAT'S YOUR EMERGENCY?

CALLER: HELP, I'M STUCK IN A MAZE AND I CAN'T GET OUT!

GAME DESCRIPTION

- In the Endless Maze game, a maze is displayed on a LED display and a joystick is used to traverse through the maze. Once the end of the maze is reached, a new maze is generated and the game continues.

MICROCHIP PIC24 MICROCONTROLLER

ANALOG 2-AXIS THUMB JOYSTICK WITH SELECT BUTTON + BREAKOUT BOARD

ADAFRUIT NEOPIXEL NEOMATRIX 8X8 - 64 RGB LED PIXEL MATRIX

LCD MODULE

ADAFRUIT 5 V USB BREAKOUT

APPLICATION DEVELOPMENT

- During the first week, two teams were formed to initially develop libraries needed for the game such as a joystick library, a LED library and a maze library of pre-constructed mazes.
- During the second week, the members of the group collaborated to complete the libraries, combine the libraries into the Forever Maze game, and prepare the presentation.

JOYSTICK STRATEGY

•Determine direction of joystick: up/down, left/right

•Communicate with Maze array to determine:

•if requested movement is allowed, e.g. no traveling through maze walls, off LED

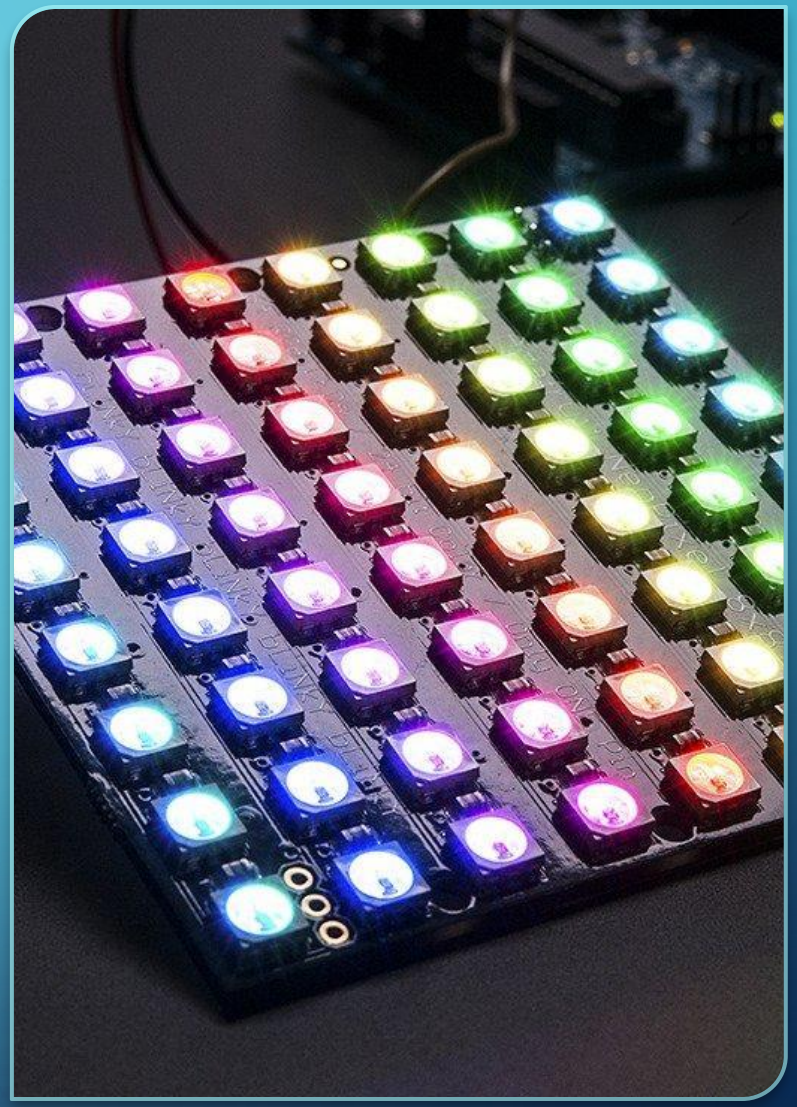
•light up square to indicate position of player

JOYSTICK IMPLEMENTATION

- joystickSetup()
- getJoystickDirection()

LED STRATEGY

- Turn on appropriate LEDs to outline the maze
- Turn on/off LEDs to indicate position of player as player moves through the maze



LED IMPLEMENTATION

- writeColor()
- updateMaze()
  - How it clears

```
;1 code total: 62.5 ns * 20 cycles = 1.25 us
;T1H: 62.5 ns * 12 cycles = 75.0 us
;20 cycles - 12 high 8 low
_write_1:
    ;call 2
    bset LATB, #7           ;high 1
    repeat #9       ;high 1
    nop             ;high 10
    bclr LATB, #7           ;low 1
    nop             ;low 1
    nop             ;low 1
    return          ;low 3 + 2 for call

;0 code total: 62.5 ns * 20 cycles = 1.25 us
;T0H: 62.5 ns * 6 cycles = .375 us
_write_0:
    ;call 2
    bset LATB, #7           ;high 1
    repeat #3       ;1
    nop             ;4
    bclr LATB, #7           ;low 1
    repeat #6       ;low 1
    nop             ;low 7
    return          ;low 3 + 2 for call
```

```
#include "bitBangHead.h"
//bit shifts right 1 bit, takes remainder of that which is our binary bit starting
//from the most significant position, checks this bit and writes in the order of
//rbg colors from R --> G --> B
void writeColor(int r, int g, int b)
{
    if(r >= 0 && r <= 255 && g >= 0 && g <= 255 && b >= 0 && b <= 255)
    {
        int shift = 7;
        int count = 8;
        while(count > 0)
        {
            if( (r >> shift) % 2 == 1)
                write_1();
            else
                write_0();
            count--;
            shift--;
        }

        shift = 7;
        count = 8;
        while(count > 0)
        {
            if( (g >> shift) % 2 == 1)
                write_1();
            else
                write_0();
            count--;
            shift--;
        }

        shift = 7;
        count = 8;
        while(count > 0)
        {
            if( (b >> shift) % 2 == 1)
                write_1();
            else
                write_0();
            count--;
            shift--;
        }

        //oneMilliSec();
    }
}
```

```
heckPlayer(int direction)

left = 3, up = 2, right = 1, down = 4, center = 5
ayerPrevRow = playerRow; //pass position to previous
ayerPrevCol = playerCol;

itch(direction)

  case 1://right
      if(playerCol == 7)//out of bounds
          break;
      else if(maze[playerRow][playerCol + 1] == 1) //wall
          break;
      else
          playerCol++; //move right
      break;

  case 2://up
  if(playerRow == 0)//out of bounds move
      break;
  else if(maze[playerRow - 1][playerCol] == 1)//wall
      break;
  else
      playerRow--; //move up
  return;

  case 3://left
      if(playerCol == 0)//out of bounds move
          break;
      else if(maze[playerRow][playerCol - 1] == 1)//wall
          break;
      else
          playerCol--; //move left
      break;

  case 4://down
      if(playerRow == 7)
          break;
      else if(maze[playerRow + 1][playerCol] == 1)
          break;
      else
          playerRow++; //move down
      break;

  default: //5 - center
      break;
```

```
void updateMaze(int direction)
{
    checkPlayer(direction);
    maze[playerPrevRow][playerPrevCol] = 0;
    maze[playerRow][playerCol] = 8;
}
```

MAZE STRATEGY

- Choosing between mazes constructed by an algorithm and pre-constructed mazes

MAZE IMPLEMENTATION

- mazeSetup()
- writeMaze()

```
heckPlayer(int direction)

left = 3, up = 2, right = 1, down = 4, center = 5
ayerPrevRow = playerRow; //pass position to previous
ayerPrevCol = playerCol;

itch(direction)

  case 1://right
      if(playerCol == 7)//out of bounds
          break;
      else if(maze[playerRow][playerCol + 1] == 1) //wall
          break;
      else
          playerCol++; //move right
      break;

  case 2://up
  if(playerRow == 0)//out of bounds move
      break;
  else if(maze[playerRow - 1][playerCol] == 1)//wall
      break;
  else
      playerRow--; //move up
  return;

  case 3://left
      if(playerCol == 0)//out of bounds move
          break;
      else if(maze[playerRow][playerCol - 1] == 1)//wall
          break;
      else
          playerCol--; //move left
      break;

  case 4://down
      if(playerRow == 7)
          break;
      else if(maze[playerRow + 1][playerCol] == 1)
          break;
      else
          playerRow++; //move down
      break;

  default: //5 - center
      break;
```

PROJECT DEMONSTRATION

Conclusion

References

- joystick tutorial (https://www.sparkfun.com/tutorials/272)
- Section 17. 10-Bit A/D Converter Manual from Microchip
- WS2812 Intelligent control LED integrated light source datasheet