Marble Maze

Team number: 14

Team Members:

Ashwin S - CB.EN.U4CCE21010

Kabil Pranav K – CB.EN.U4CCE21028

Pranav Jayakrishna – CB.EN.U4CCE21046

Yathis kumar D – CB.EN.U4CCE21079

Abstract:

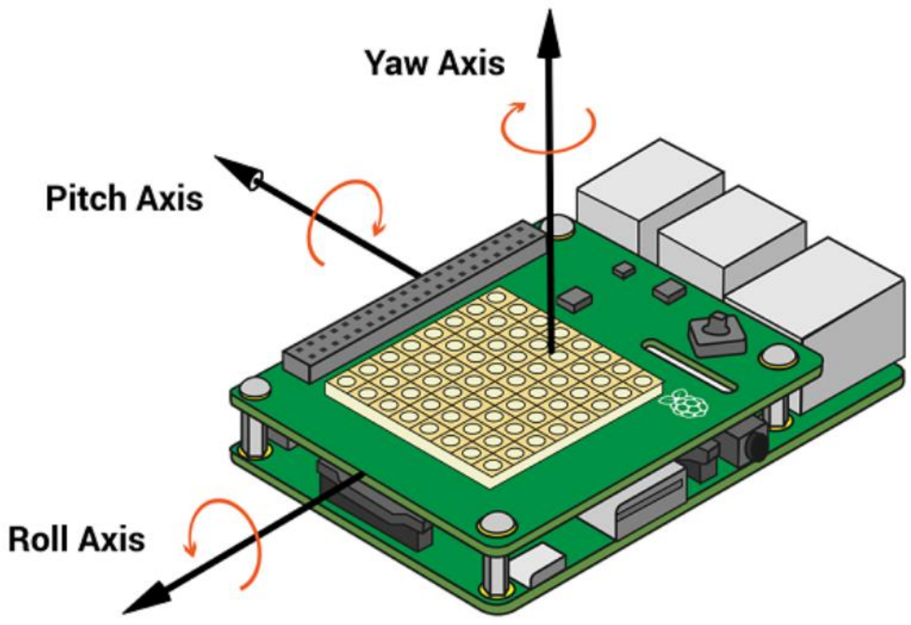
Players navigate a virtual marble through mazes by tilting the Raspberry Pi, leveraging the Sense HAT's accelerometer data. The objective is to reach the green endpoint while avoiding red obstacles. Each level has a time limit, and completion leads to the next level. The game is visually displayed on the Sense HAT's LED matrix, providing an interactive and educational Python programming experience on the Raspberry Pi platform.

Code:

```
from sense_hat import SenseHat

from time import sleep, time


sense = SenseHat()

sense.clear()


r = (255, 0, 0)

g = (0, 255, 0)

b = (0, 0, 0)

w = (255, 255, 255)


def display_level(level):

    sense.show_message(f'Level {level}')
```

```
def play_level(maze):
    global game_over, x, y
    game_over = False
    start_time = time()
    lost = False

    while not game_over and not lost:
        move_marble()
        check_win(x, y)
        maze[y][x] = w
        sense.set_pixels(sum(maze, []))
        sleep(0.05)
        maze[y][x] = b

        elapsed_time = time() - start_time
        if elapsed_time > 10:
            sense.show_message('You Lose')
            lost = True
            return lost



    if not lost:
        sense.show_message('You Win')

def move_marble():
    global x, y

    pitch = sense.get_orientation()['pitch']
    roll = sense.get_orientation()['roll']

    new_x = x
    new_y = y

    if 1 < pitch < 179 and x != 0:
        new_x -= 1
    elif 359 > pitch > 179 and x != 7:
        new_x += 1

    if 1 < roll < 179 and y != 7:
        new_y += 1
    elif 359 > roll > 179 and y != 0:
        new_y -= 1
    x, y = check_wall(x, y, new_x, new_y)

def check_wall(x, y, new_x, new_y):
    if maze[new_y][new_x] != r:
        return new_x, new_y
    elif maze[new_y][x] != r:
        return x, new_y
    elif maze[y][new_x] != r:
        return new_x, y

    return x, y

def check_win(x, y):
    global game_over
    if maze[y][x] == g:
        game_over = True



levels = [
    [[r, r, r, r, r, r, r, r],
 [r, b, b, b, b, b, b, r],
 [r, r, r, b, r, b, b, r],
 [r, b, r, b, r, r, r, r],
 [r, b, b, b, b, b, b, r],
 [r, b, r, r, r, r, b, r],
 [r, b, b, r, g, b, b, r],
 [r, r, r, r, r, r, r, r]],

[[b, r, r, r, r, r, b, r],
 [r, b, r, r, r, b, b, r],
 [r, r, b, r, b, r, r, r],
 [r, r, r, b, r, r, b, g],
 [r, r, b, r, b, r, b, r],
 [r, b, r, r, r, b, r, r],
 [b, r, r, r, b, r, b, r],
 [r, b, b, b, b, b, b, b]],

[[b, r, r, r, r, r, r, b],
 [r, b, b, r, r, b, b, r],
 [r, r, b, r, r, b, b, r],
 [b, b, b, b, b, b, r, r],
 [b, b, b, b, b, b, r, g],
 [r, r, b, r, r, b, b, r],
 [r, b, b, r, r, b, b, b],
 [b, r, b, b, b, b, b, b]],

[[r, b, b, b, b, r, r, r],
 [r, b, r, b, r, b, b, r],
 [b, r, r, b, r, b, b, r],
 [r, r, r, r, b, r, b, r],
 [b, r, r, b, r, b, r, r],
    [b, r, b, r, r, r, r, r],
    [r, b, r, b, r, b, r, g],
    [b, r, r, b, b, r, b, b]],
]
p=0
for i, maze in enumerate(levels, start=1):
   display_level(i)
   x = 1
   y = 1
   k=play_level(maze)
   if k == True:
      break
   p=i
sense.show_message("Score:"+str(p))
```

Circuit Diagram:



Working Principle:

1) Initialization: The script sets up the necessary environment by importing libraries, initializing the Sense HAT, and defining color variables.

2) Display and Play Functions: Functions are created to display levels and control gameplay. The play function manages the virtual marble's movement using pitch and roll sensor data.

3) Move Marble Function: Determines the marble's new position based on sensor values, adjusting for valid moves and handling collisions with maze walls.

4) Check Wall Function: Ensures the marble doesn't pass through walls by validating the new position against the maze layout.

5) Check Win Function: Identifies if the marble reaches the goal, declaring victory when the goal is reached.

6) Level Definitions: Predefined levels are represented as matrices, defining unique maze layouts. Additional levels can be easily added.

7) Level Iteration: The script iterates through levels, displaying, playing, and updating the score. If the player loses a level, the loop breaks, and the final score is displayed.

8) Score Display: The Sense HAT shows a message at the end, indicating the player's final score based on completed levels.

Output:

https://youtube.com/shorts/ONrDdeA5pcg