UPPSALA
UNIVERSITET

Report for 1DT086

Project Title - Implementation of Conway's game of life on 8x8 LED Matrix

Group 05

Nusrat Hossain Aneysha Datta Deepa Gopi

December 19, 2019

1 Introduction

This project is based on the cellular automation which initially takes an input for live cells and then creates the simulation for future generations based on certain rules.

2 Conway's Game Rules

In this Conway's game of life we followed certain rules. Those are:

- If any dead cell has three live cells among its eight neighbours it becomes alive in the next generation.
- If any live cell has two or three neighbouring live cells it will stay alive in the next generation.
- If any live cell has less than 2 or more than 3 neighbouring live cells, it will die.
- All other cells are dead.

3 Conway's Game Features(as per the given instructions)

The features for Conway's game are given below:

- Dead cells are displayed as LEDs turned off.
- Alive cells are displayed with green LEDs.
- The cursor is displayed as a single red LED (but is not displayed once the simulation starts).
- The game starts with a blank board (all cells are "dead"), with the "cursor" at the pixel at the top left corner of the LED matrix.
- Before the simulation starts, the player can move the cursor around the LED matrix with the joystick, and place as many "alive" cells as she pleases. An alive cell is placed at the current position of the cursor by pushing the joystick down.
- Once the player pushes the joystick down three times in succession, without having moved the cursor in between, the simulation starts, and the cursor goes away.

- The simulation then proceeds with one new generation/tick once every second, and ends only once the player presses the joystick down again. When it ends, the program terminates.
- Cursor is moving in a wraparound fashion. The borders of the matrix are the end points.
- When the user takes the cursor back on a selected green cell and presses down again the live cell is unselected.(extra feature)

4 Requirements for the project

- Sense Hat
- HDMI cable
- Monitor
- Power Cable
- Raspberry pi

5 Programming environment

Here in this project Python is used as the main programming language.

6 Description of the code

We have used one generic function def check_if_green that is being called throughout the program by several other functions.

```
ddef check_if_green(position_x,position_y,all_pts):
  is_there = False
  if 0<= position_x <= 7 and 0<= position_y <= 7:
    for i in range(len(all_pts[0])):
      if position_x == all_pts[0][i]:
        if position_y == all_pts[1][i]:
          is_there = True
  return is_there
```

This function is checking all the points on the 8x8 LED matrix to see if that cell is already present within the set of all_pts or not.It is first iterating the loop for the rows .If the row is present, it checks for the corresponding point for column from the matrix values.If it is present, the is_there flag is reinitialized to True and returned back to the calling function.

For the below inputs from the user

```
mySense.stick.direction_left = cursor_left
mySense.stick.direction_right = cursor_right
mySense.stick.direction_down = cursor_down
mySense.stick.direction_up = cursor_up
mySense.stick.direction_middle = single_press
```

We have defined some functions. Snippets with descriptions are described below:

```
if event.action != ACTION_RELEASED:
    global cursor_position_x, cursor_position_y, start_pts,

# variable for resetting the number of presses to zero so that it can be
    determined if there are 3 presses on the same point.

    press_count = 0

# if the present point is not green, it will be visualized as black,if it is
    green, there will be no change.

    if not check_if_green(cursor_position_x,cursor_position_y,start_pts):
      mySense.set_pixel(cursor_position_x,cursor_position_y,black)
    cursor_position_x = cursor_position_x - 1

# wrapping around the position

    if cursor_position_x < 0 :
      cursor_position_x = 7

# if the new point is green, the cursor will not be visible, if not green, it
     will appear as a red dot.

    if not check_if_green(cursor_position_x,cursor_position_y,start_pts):
      mySense.set_pixel(cursor_position_x,cursor_position_y,red)
```

We defined a function called single_press(event) which is given below:

```
def single_press(event):

  if event.action != ACTION_RELEASED:
    global cursor_position_x, cursor_position_y,start_pts, press_count,
        start_play, total_run
    if start_play == 0:
      press_count = press_count + 1
      if press_count == 1:


        if check_if_green(cursor_position_x,cursor_position_y,start_pts) :
           for i in range(len(start_pts[0])):

            if cursor_position_x == start_pts[0][i]:
              if cursor_position_y == start_pts[1][i]:

                del start_pts[0][i]
                del start_pts[1][i]
                mySense.set_pixel(cursor_position_x,cursor_position_y,red)
                break
        else:
          mySense.set_pixel(cursor_position_x,cursor_position_y,green)
          start_pts[0].append(cursor_position_x)
          start_pts[1].append(cursor_position_y)
      if press_count == 3:

        play_game(start_pts)
    else:
      mySense.clear()
      total_run = 1
```

Here it checks wheather there is single press or multiple preses from the users

and accordingly increses the press_count variable. if there are three consecutive presses the simulaton starts.We have given a additional feature. If the user goes back on a selected live cell and presses it,it removes the cell from the selection.if the user presses once on a single cell,it turns it green.

The function find_neig_count helps in recognizing the neighbouring green cells around any particular point.The number of green cells is then appended to n_count

We used another function play_game which runs recursively and continues the simulation.Here we declare two variables death_list and birth_list .death_list stores the index of the dead cells and birth_list stores the coordinates of the cells which will be live for the next generation. To check the number of neighbouring live cells around a point it calls the function find_neig_count and according to the n_count value it appends the death and birth list variables. The birth list finally contains coordinates of dead cells which will be born in the next generation and the cells which will remain alive.It then lights up this cells every 1 second

7 Testing

After implementation of the code we tried testing the code based on various patterns available from the information sources readily available online (mainly Wikipedia).We were successfully able to get the below patterns on the 8x8 LED matrix.

Still Patterns:

- Block
- Beehive
- Boat
- Loaf
- Tub

Oscillating Patterns:

- Blinker
- Toad
- Beacon

Spaceships: Glider

8 Challenges and Improvisations

We are beginners in using python so it took a lot of effort to implement all the features.

- While the simulation started the joystick refused to take input from user as there was recursive looping of the function play_game. We later realised that the object event was getting referred each time for the user input which we were trying to create within the simulation. Hence we modified it to event1. This sorted out the problem of ending the game.
- After the game stopped, we initially had to press a Ctrl+C to get the prompt. It took us some time to realise that the endless loop was not terminating even after the game stops.
- The program has a discrepancy that when the glider reaches the ends of the matrix, these are treated as end points and the index goes out of range. We still need to improvise on this.
- The sleep time is defined to 2 second between presses. If the user takes a long time to press, it is counted as a single press again and the previous presses are missed. Hence, the user has to press a bit faster on a cell for the simulation to start.

9 Conclusion

A lot of teamwork is involved in this project and each of us got an exposure to a new programming language Python. It also enhances our troubleshooting skills which are a very big achievement in the long run.

References

[1] Youtube
`https://www.youtube.com/watch?v=K-JdVSdOaLE`

[2] Youtube
`https://www.youtube.com/watch?v=ouipbDkwHWA`

[3] Pythonhosted.
`https://pythonhosted.org/sense-hat/api/`

[4] Sense-emu
`https://sense-emu.readthedocs.io/en/v1.0/examples.html`

[5] Github
`https://github.com/astro-pi/beta-testing/issues/29`

[6] Wikipedia
https://en.wikipedia.org/wiki/Conway%27s_Game_of_Life