CS210-Mini Project FPGA

Adithya Ravi Rathod

2101CS04

❖ Overview:

The Ring Puzzle game is a single person game designed entirely for DE1-SoC board in C language. This game requires the DE1-SoC board and external VGA display.

The motive of this game is to *rotate* the three gears *clockwise* and place all the RED dots over the triangle at the centre. The keys assigned for the rotation of the gears are:

G -> left gear

H -> right gear

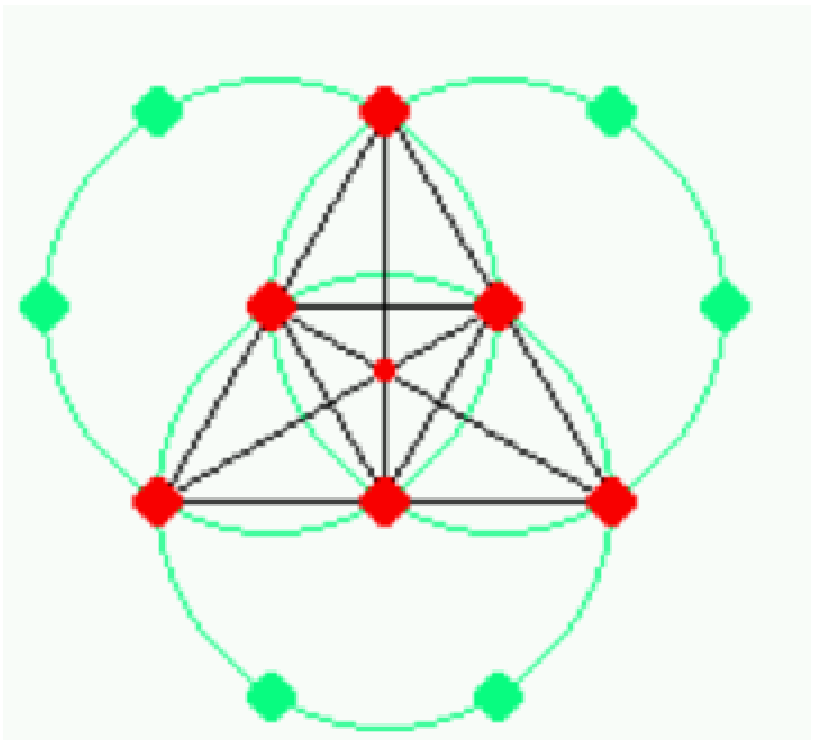
B-> bottom gear

*This is the desired pattern by the rotation of the

gears.

*Centre do in fixed and just for

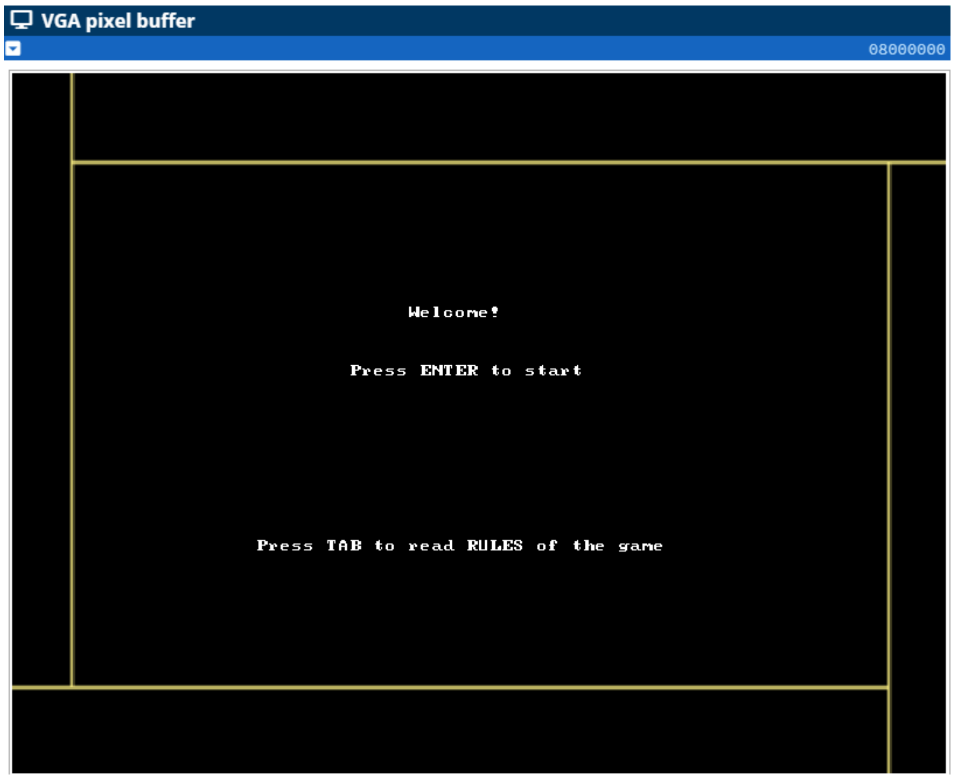
Aesthetic reasons :>



❖ Demonstration:

First the home screen loads and Welcomes is the user with two options either to start the game by pressing ENTER or read the rules using the TAB key.

The Home Screen:

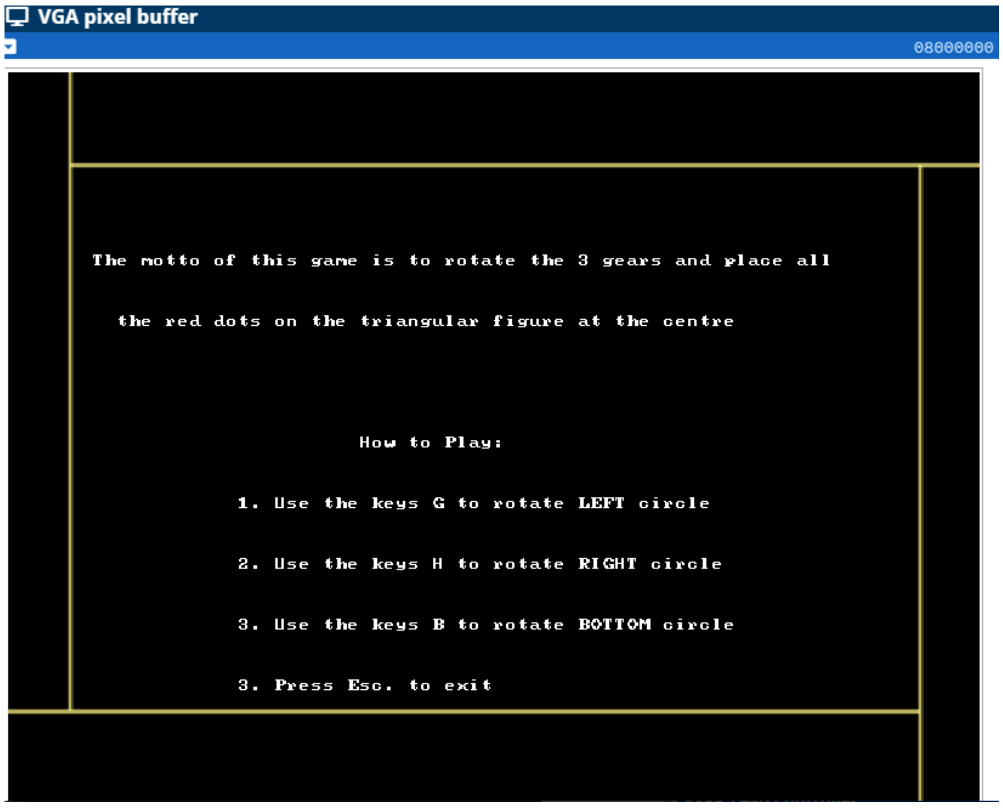


Welcome!

Press ENTER to start

Press TAB to read RULES of the game

The Rules page:



The motto of this game is to rotate the 3 gears and place all

the red dots on the triangular figure at the centre

How to Play:

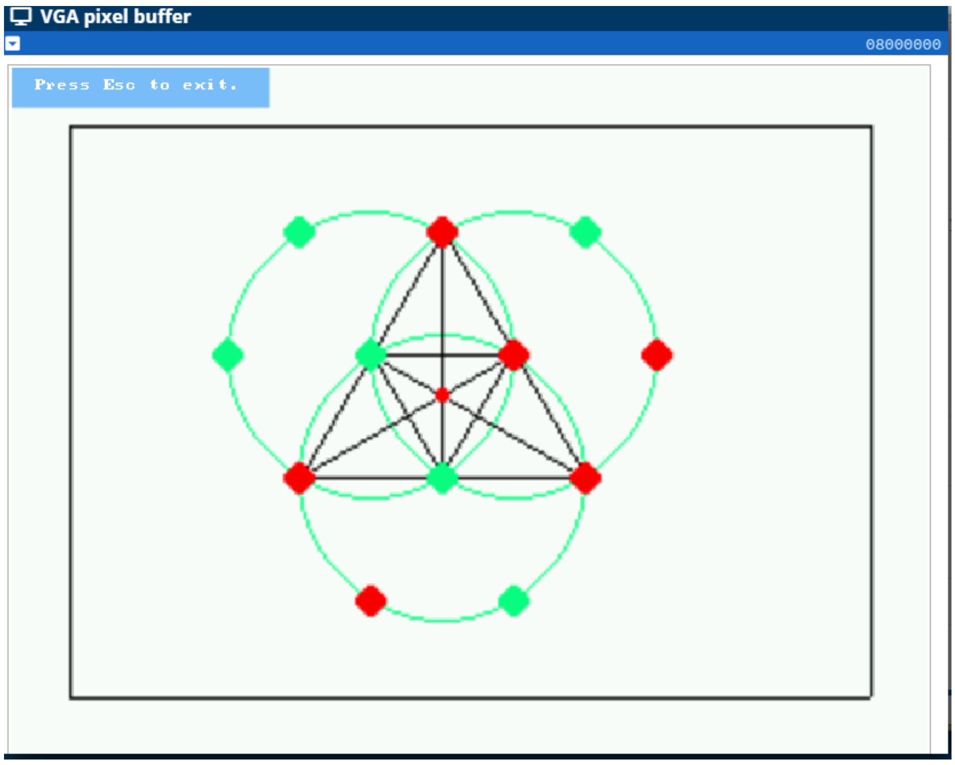
1. Use the keys G to rotate LEFT circle

2. Use the keys H to rotate RIGHT circle

3. Use the keys B to rotate BOTTOM circle

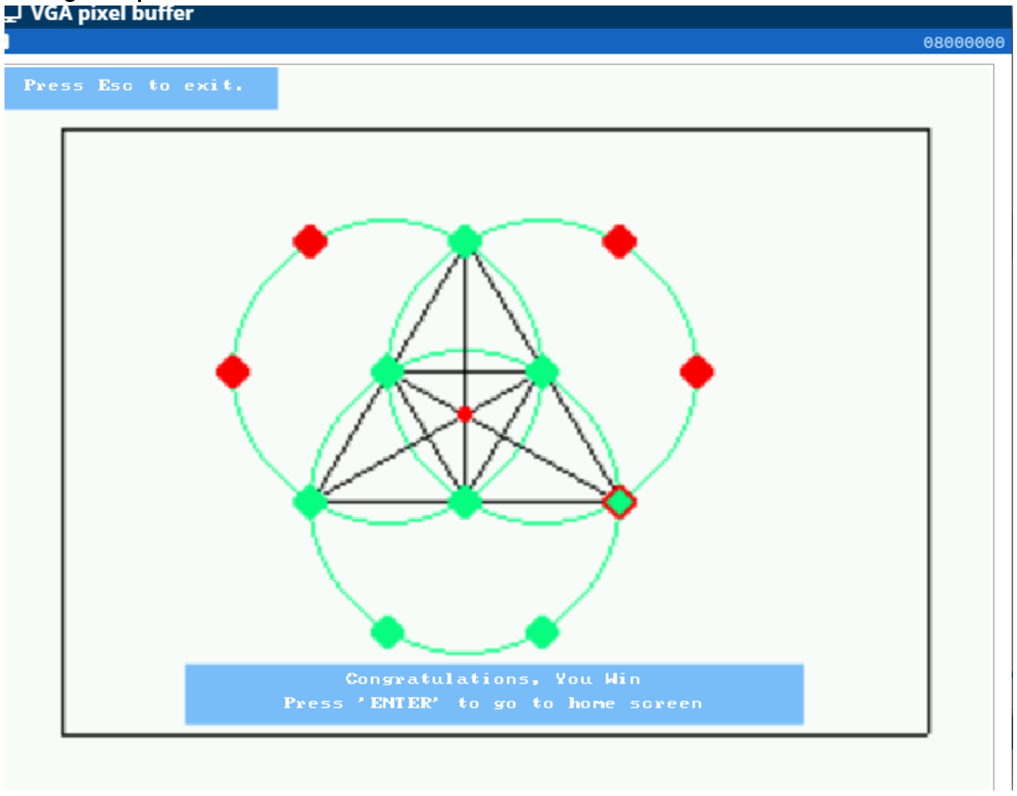
3. Press Esc. to exit

The Puzzle Screen:



Press Esc to exit.

*A randomized order of 6 RED and 6 GREEN dots is generated every time you start solving the puzzle



Press Esc to exit.

Congratulations, You Win
Press 'ENTER' to go to home screen

*Snapshot of the blinking effect

*After the puzzle is solved, a blinking effect mark the success of the game and user is prompted to go to the home screen which again restarts the game from home screen upon hitting the ENTER key.

❖ Contribution

- ➢ The logic of the game is entirely my work.
- ➢ Design of the code and debugging is conducted by me.
- ➢ I have used some of the algorithms like Bresenham' s Algorithm to draw lines and circles on the VGA output .
- ➢ Also, the code for configuring key bindings is already present online.

❖ Code Explanation

Code begins with defining JTAG_UART_BASE to location (0xFF201000) and JTAG_UART_CONTROL to location (0xFF201000 + 4).

We have the following functions in the code in order:

```
1. writePixel(int x, int y, short color)
```
a. writes a pixel with coordinate (x, y) with 'colour'.

```
2. readPixel(int x, int y)
```
a. reads the colour of the pixel at (x, y).

```
3. writeChar(int x, int y, char c)
```
a. writes into the character buffer the character 'c' at point (x, y).

```
4. get_jtag(volatile int *JTAG_UART_ptr)
5. put_jtag(volatile int *JTAG_UART_ptr, char c)
6. writeString(int x, int y, char s[])
```
a. To write string into the character buffer. Just Looping the write char over the character array s[ ].

```
7. clearScreen()
```
a. write the entire screen of 320x240 pixel with black colour
b. And also write all the characters in character buffer 80x60

```
8. helpCircle(int xc, int yc, int x, int y, short c)
9. drawCircle(int xc, int yc, int r, short colour)
```
a. Two functions which implement Bresenham' s Algorithm for drawing circles on the VGA display.

```
10. drawLine(int x1, int y1, int x2, int y2, short colour)
```
a. Bresenham' s Algorithm for drawing a line from (x1, y1) to (x2, y2) in colour specified by 'colour'.

```
11. outerStructure(int xc, int yc, int r, short col, short col2)
```

a. Draws the circle boundary (a skeletal structure).

```
12. colorFill(int xc, int yc, int r, int x)
```

a. Fills the area of a circular region from centre to the radius r.

```
13. fillMyArray(int arr[])
```

a. initializing the colour of the dot at the start of the game.

```
14. boundary()
```

a. Boundary for the puzzle screen.

```
15. swap(int *a, int *b)
16. randomize(int arr[], int n)
```

a. randomizing the dots after every play.

```
17. rotate1(int a[12], int b[12])
```

a. The organization of the dots after the LEFT gear is moved.

```
18. rotate2(int a[12], int b[12])
```

a. The organization of the dots after the Right gear is moved.

```
19. rotate3(int a[12], int b[12])
```

a. The organization of the dots after the Right gear is moved.

```
20. charblock()
21. charblock2()
```

a. just background for the screen puzzle because white background won't show strings in character buffer.

```
22. blink()
```

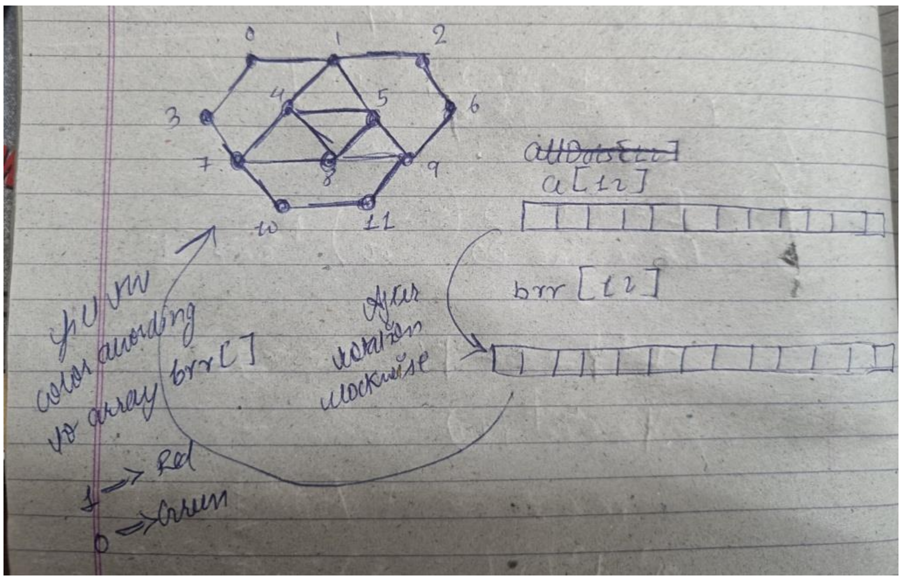
a. To create a blinking animation after the game is cleared.

```
23. ringPuzzle()
```

a. The main game and all logics.
b. An array of 12 elements called alldots[12] is initialized with six 0s and six 1s
c. This array is the randomized and fillMyArray() is called to fill the dots in the skeletal with appropriate colour (1 -> RED and 2 -> GREEN).
d. Setting the while infinite loop and set the key bindings over the host PC to rotate the gears and the game cleared logic.

```
24. homeScreen()
25. rules()
26. main()
```

❖ Logic on Paper :)



0
1
2
3
4
5
6
7
8
9
10
11
~~alldots[12]~~
a[12]
brr[12]
After rotation clockwise
fill color according to array brr[]
1 → Red
0 → Green