TankGo!

CSEE 4840 Embedded Systems

Jiayi Wang (jw4462)

Xuanbo Xu (xx2440)

Yizhi Wang (yw4174)

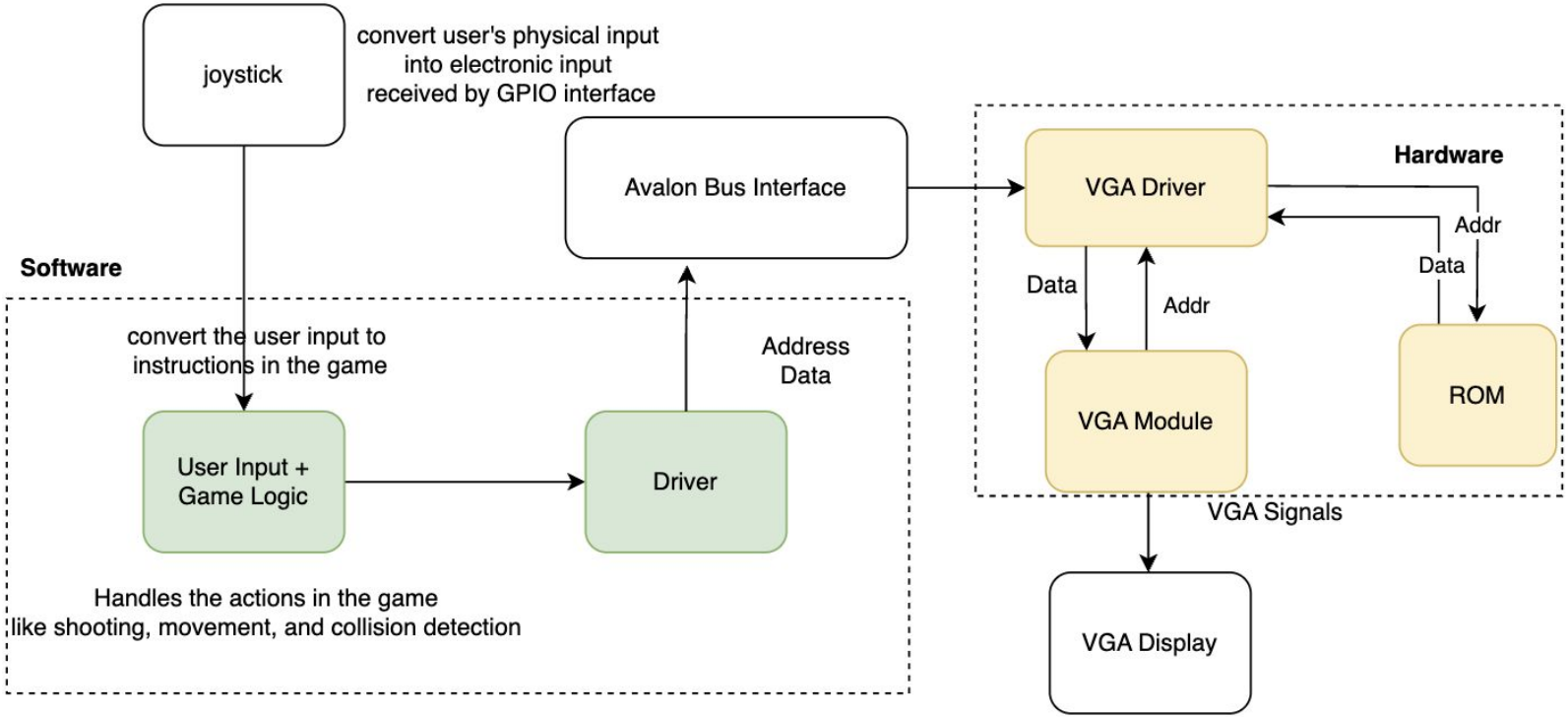
Yiyang Peng (yp2655)

May 10, 2024

Overview and Rules

- There exists three maze maps.
- Two players move tank around a maze and shoot bullet at one another.
- Players use the ⬆️⬇️➡️⬅️ buttons on joysticks to move the tank
- Players shoot with the attack 🅰️button.
- Ball bounces off walls 15 times then disappears if no tank was hit.
- If a tank gets hit by opponent or itself, it loses HP.
- When a tank has no HP, game over.

System Block Diagram



Avalon Bus Interface

Address /Bits	7	6	5	4	3	2	1	0
0	Tank 1	Score	0 - 15		Tank 2	Score	0 - 15	
1								End
2	Tank 1	Location	X	Coord	0 - 39	8 bits		
3	Tank 1	Location	Y	Coord	0 - 39	8 bits		
4	Tank 2	Location	X	Coord	0 - 39	8 bits		
5	Tank 2	Location	Y	Coord	0 - 39	8 bits		
6						Tank 1	8 Direct	3 bits
7						Tank 2	8 Direct	3 bits
8	Bullet 1	Location	X	Coord	0 - 39	8 bits		
9	Bullet 1	Location	Y	Coord	0 - 39	8 bits		
10	Bullet 2	Location	X	Coord	0 - 39	8 bits		
11	Bullet 2	Location	Y	Coord	0 - 39	8 bits		

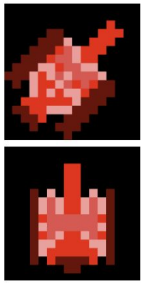
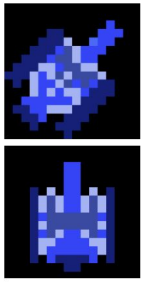
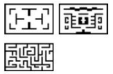
If bullet loc == 0b00000000, Bullet does not display.

Tank 1 HP = 16 - Tank 2 score

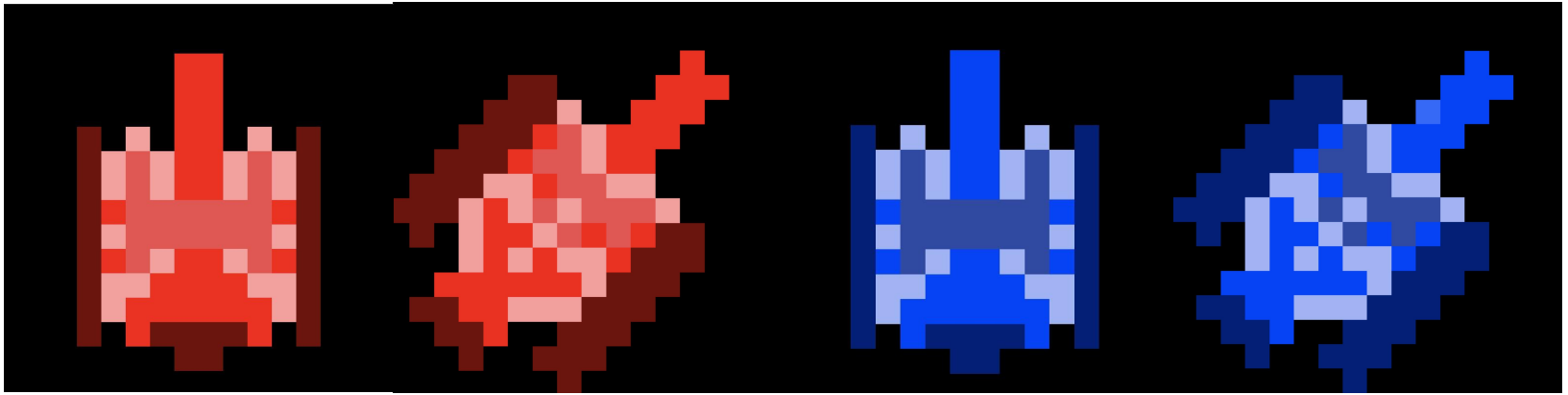
When game over, End is high.

Memory

Total:28568

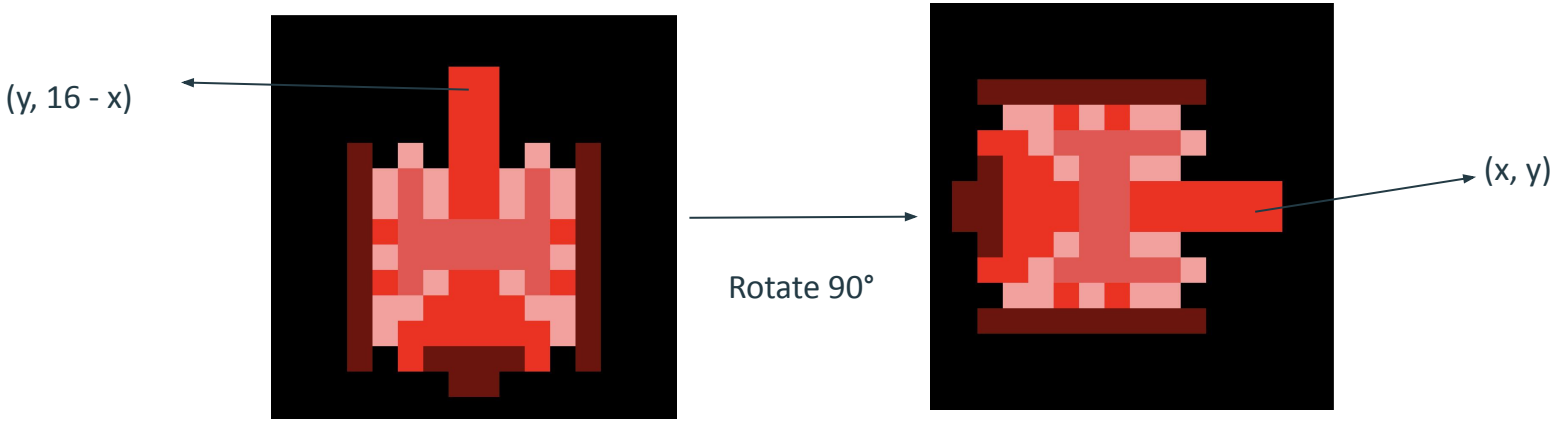
Category	Graphics	Size(bits)	# of images	Total Size(bits)
Tank 1		16*16	2	12288
Tank 2		16*16	2	12288
Text 1		48*8	1	384
Text 2		60*8	1	480
Text 3		16*8	1	128
Map		40*25	3	3000

Graphics



We used matrix translations on mifs to achieve 8 directions with only four sprites in ROM for two tanks.

Graphics



We used matrix translations on mifs to achieve 8 directions with only four sprites in ROM for two tanks.

Joystick

Constant	Constant	Constant	h_dirc	v_dirc	XYAB	Other keys

- Each controller communicates using the 7 byte protocol above⬆️
- The three constants are all 255, representing the protocol 0 being used in these joysticks. The h_dirc and v_dirc are the directional inputs from the joysticks.
- **h_dirc** changes to 0 when left arrow is pressed and to 255 when right arrow is pressed. Similarly, **v_dirc** changes to 0 when up arrow is pressed and to 255 when down is pressed.
- **XYAB:** Different integer values represent different combinations of these button being pressed.
- Other keys were not used in our project.

Joystick

Spin left
When h_dirc = 0

Drive Forward
When v_dirc = 0
Also for selecting the next map

Reverse
When v_dirc = 255
Also for selecting the last map

Spin right
When h_dirc = 255

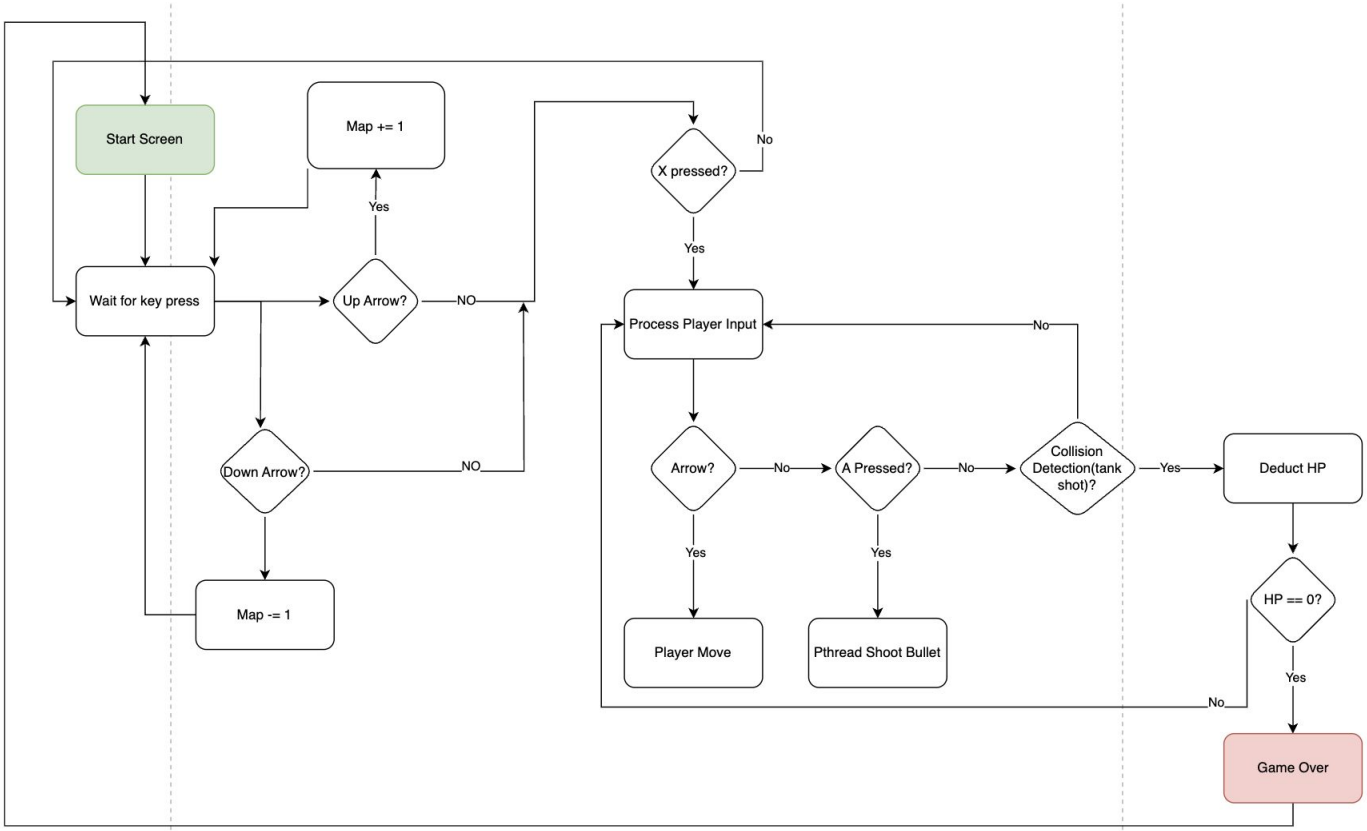
X is for select map

When XYAB = 31

A for shoot bullet

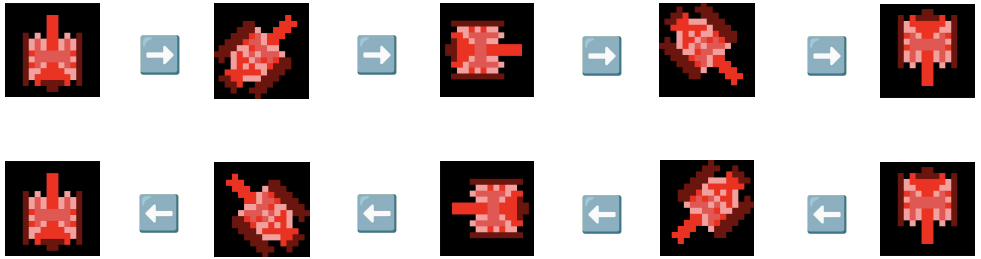
When XYAB = 47

Game Logic


Start Screen
Map += 1
X pressed?
No
Yes
Wait for key press
Up Arrow?
NO
Process Player Input
No
Down Arrow?
NO
Arrow?
No
A Pressed?
No
Collision Detection(tank shot)?
Yes
Deduct HP
Map -= 1
Player Move
Pthread Shoot Bullet
HP == 0?
Yes
No
Game Over

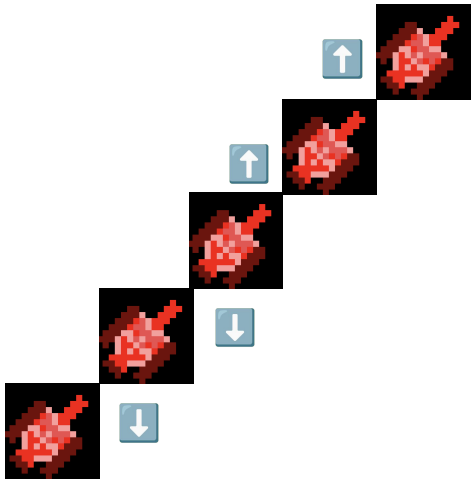
Tank Movement

Four arrow buttons but 8 directions.



etc.

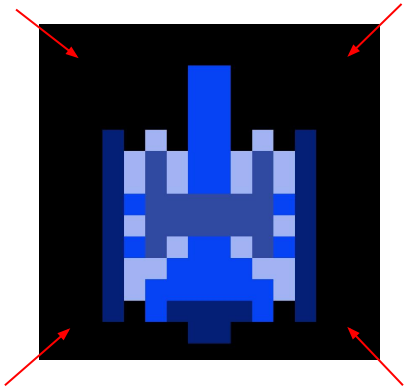
etc.



Collision Detection (Tank with Map)

(Tank1_x, Tank1_y)

(Tank1_x + 3, Tank1_y)



(Tank1_x , Tank1_y - 3)

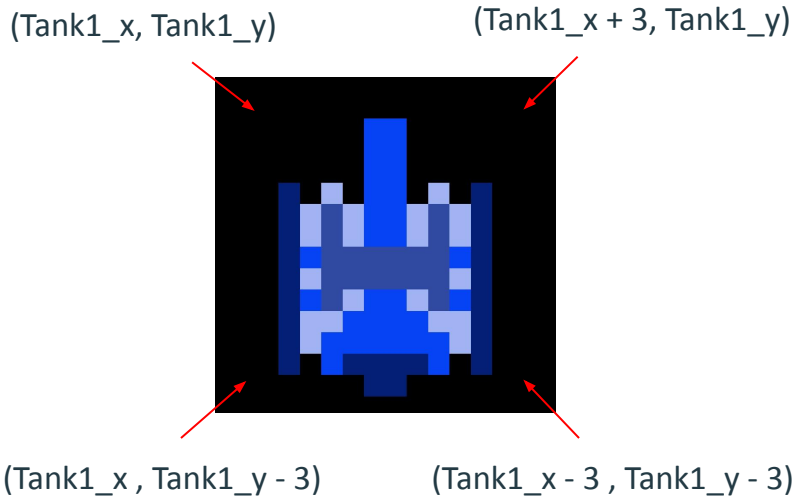
(Tank1_x - 3 , Tank1_y - 3)

```
int index = x + y * 8;
int index1 = x+3 + y * 8;
int index2 = x + (y-3) * 8;
int index3 = x-3 + (y-3) * 8;

if (index >= 0 && index < 64 && (map[index]
== 1|| map[index1] == 1 || map[index2] == 1
|| map[index3] == 1)) {
   return 1;
} else {
   return 0;
}
```

```
Int map1 [64] = {
1, 1, 1, 1, 1, 1, 1, 1,
1, 0, 0, 1, 0, 0, 0, 1,
1, 0, 0, 0, 0, 0, 0, 1,
1, 0, 1, 1, 1, 1, 0, 1,
1, 0, 1, 0, 0, 0, 0, 1,
1, 0, 1, 0, 1, 1, 1, 1,
1, 0, 1, 0, 0, 0, 0, 1,
1, 1, 1, 1, 1, 1, 1, 1}
```

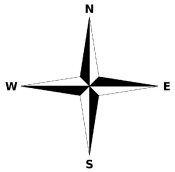
Collision Detection (Bullet with Tank)




```
if bullet_x between Tank1_x and Tank1_x - 3,
AND bullet_y between Tank1_y and Tank1_y - 3) {
    return 1;
}
    return 0;
}
```

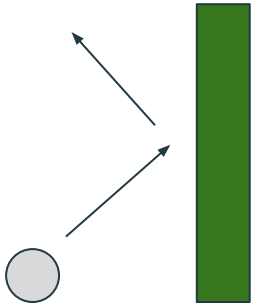
Collision Detection (Bullet with Wall)

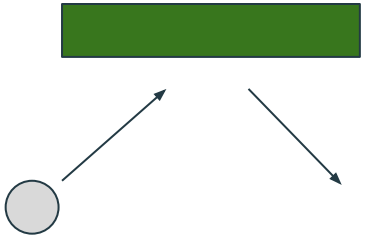
Initially, bullet direction = tank direction.

If direction is 

, if bullet_wall_collision() == 1, N ➡️S, E ➡️W, S ➡️N, W ➡️E.

If direction is ex. NE, two possibilities:





If bullet_x+1 collides, NE ➡️NW.
If bullet_y+1 collides, NE ➡️SE.