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 A 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

- **Joystick**: Converts user's physical input into electronic input received by GPIO interface.
- **Avalon Bus Interface**
- **VGA Driver**
- **VGA Module**
- **ROM**
- **VGA Display**

**Software**

- **User Input + Game Logic**: Converts user input to instructions in the game.

**Hardware**

- **VGA Signals**
- **Data**
- **Addr**

**Description**

- Handles the actions in the game like shooting, movement, and collision detection.
If bullet loc == 0b00000000, Bullet does not display.

Tank 1 HP = 16 - Tank 2 score

When game over, End is high.

<table>
<thead>
<tr>
<th>Address /Bits</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Tank 1</td>
<td>Score</td>
<td>0 - 15</td>
<td>Tank 2</td>
<td>Score</td>
<td>0 - 15</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>End</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Tank 1</td>
<td>Location</td>
<td>X</td>
<td>Coord</td>
<td>0 - 39</td>
<td>8 bits</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Tank 1</td>
<td>Location</td>
<td>Y</td>
<td>Coord</td>
<td>0 - 39</td>
<td>8 bits</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Tank 2</td>
<td>Location</td>
<td>X</td>
<td>Coord</td>
<td>0 - 39</td>
<td>8 bits</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Tank 2</td>
<td>Location</td>
<td>Y</td>
<td>Coord</td>
<td>0 - 39</td>
<td>8 bits</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Tank 1</td>
<td>8 Direct</td>
<td>3 bits</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>Tank 2</td>
<td>8 Direct</td>
<td>3 bits</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>Bullet 1</td>
<td>Location</td>
<td>X</td>
<td>Coord</td>
<td>0 - 39</td>
<td>8 bits</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>Bullet 1</td>
<td>Location</td>
<td>Y</td>
<td>Coord</td>
<td>0 - 39</td>
<td>8 bits</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>Bullet 2</td>
<td>Location</td>
<td>X</td>
<td>Coord</td>
<td>0 - 39</td>
<td>8 bits</td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>Bullet 2</td>
<td>Location</td>
<td>Y</td>
<td>Coord</td>
<td>0 - 39</td>
<td>8 bits</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Memory

**Total:** 28568

<table>
<thead>
<tr>
<th>Category</th>
<th>Graphics</th>
<th>Size (bits)</th>
<th># of images</th>
<th>Total Size (bits)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tank 1</td>
<td><img src="image1.png" alt="Tank1 Image" /></td>
<td>16*16</td>
<td>2</td>
<td>12288</td>
</tr>
<tr>
<td>Tank 2</td>
<td><img src="image2.png" alt="Tank2 Image" /></td>
<td>16*16</td>
<td>2</td>
<td>12288</td>
</tr>
<tr>
<td>Text 1</td>
<td><img src="image3.png" alt="Text1 Image" /></td>
<td>46*8</td>
<td>1</td>
<td>384</td>
</tr>
<tr>
<td>Text 2</td>
<td><img src="image4.png" alt="Text2 Image" /></td>
<td>60*8</td>
<td>1</td>
<td>480</td>
</tr>
<tr>
<td>Text 3</td>
<td><img src="image5.png" alt="Text3 Image" /></td>
<td>16*8</td>
<td>1</td>
<td>128</td>
</tr>
<tr>
<td>Map</td>
<td><img src="image6.png" alt="Map Image" /></td>
<td>40*25</td>
<td>3</td>
<td>3000</td>
</tr>
</tbody>
</table>
Graphics

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

<table>
<thead>
<tr>
<th>Constant</th>
<th>Constant</th>
<th>Constant</th>
<th>h_dirc</th>
<th>v_dirc</th>
<th>XYAB</th>
<th>Other keys</th>
</tr>
</thead>
</table>

- 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

Drive Forward
When \( v\_dirc = 0 \)
Also for selecting the next map

Reverse
When \( v\_dirc = 255 \)
Also for selecting the last map

Spin left
When \( h\_dirc = 0 \)

Spin right
When \( h\_dirc = 255 \)

X is for select map
When \( XYAB = 31 \)

A for shoot bullet
When \( XYAB = 47 \)
Game Logic

1. Start Screen
2. Wait for key press
3. Map := 1
4. Up Arrow?
   - Yes
   - No (Map := 1)
5. Down Arrow?
6. X pressed?
   - Yes
   - No
7. Process Player Input
   - Arrow?
     - No
     - A Pressed?
       - No
       - Collision Detection (tank shot?)
         - Yes
         - Deduct HP
         - HP == 0?
           - Yes
           - Game Over
         - No
         - Player Move
         - Pithead Shoot Bullet
   - No
8. Game Over
Tank Movement

Four arrow buttons but 8 directions.

➡️
⬅️
⬆️
⬇️

e tc.

e tc.

e tc.
Collision Detection (Tank with Map)

\[(\text{Tank}_1_x, \text{Tank}_1_y)\] \hspace{1cm} \[(\text{Tank}_1_x + 3, \text{Tank}_1_y)\]

\[(\text{Tank}_1_x, \text{Tank}_1_y - 3)\] \hspace{1cm} \[(\text{Tank}_1_x - 3, \text{Tank}_1_y - 3)\]

```c
int map1[64] = {
    1, 1, 1, 1, 1, 1, 1, 1,
    1, 0, 0, 0, 0, 0, 0, 0,
    1, 1, 1, 1, 1, 0, 1,
    1, 0, 1, 0, 0, 0, 0, 0,
    1, 1, 0, 1, 0, 1, 1, 1,
    1, 0, 1, 0, 0, 0, 0, 0,
    1, 0, 1, 1, 1, 1, 1, 1,
};

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;
}
```
Collision Detection (Bullet with Tank)

```c
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 \[\begin{array}{c}
\text{N} \\
\text{E} \\
\text{S} \\
\text{W}
\end{array}\], if \text{bullet\_wall\_collision}() == 1, N → S, E → W, S → N, W → E.

If direction is ex. NE, two possibilities:

If \text{bullet\_x+1} collides, NE → NW.

If \text{bullet\_y+1} collides, NE → SE.