Real-Time Subway Data

Displays location of trains on the subway map in real-time.
Goals

- Pan screen using hardware display buffer
- Animate train on the map
- Continuously update one line with live locations
- Support switching to different lines

Constraints

- Limited on-board storage space
- Small screen size with a big map
- Must poll an API in real time
Implementation

Software

- Takes in train arrival times and keyboard input
- Computes position based on arrival time
- Gives positions for train sprites to be created

Hardware

- Draws to the screen
- Updates sprite positions
Graphics

Two layers of graphics:

- Background with subway lines and stations
- Sprites generated through software

Display 500x425
Status bar: 300x50
Sprites

Register 1: Update Sprite in Table

Type 0: 4-bit color (16 colors) for trains

<table>
<thead>
<tr>
<th>(MSB)</th>
<th>31:24</th>
<th>23</th>
<th>22:21</th>
<th>20:16</th>
<th>15:13</th>
<th>12:8</th>
<th>7:5</th>
<th>4:0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Table Offset</td>
<td>type</td>
<td>padding</td>
<td>New Sprite #</td>
<td>padding</td>
<td>width</td>
<td>padding</td>
<td>height</td>
</tr>
</tbody>
</table>

Type 1: 1-bit color for letters

<table>
<thead>
<tr>
<th>(MSB)</th>
<th>31:24</th>
<th>23</th>
<th>22:16</th>
<th>15:13</th>
<th>11:8</th>
<th>7:4</th>
<th>3:0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Table Offset</td>
<td>type</td>
<td>New Sprite #</td>
<td>padding</td>
<td>red</td>
<td>green</td>
<td>blue</td>
</tr>
</tbody>
</table>
Constraint #1: Limited On-Chip Storage

- Map is 1500x1814 pixels
  - Too big to store with color for each pixel
- **Solution: Run Length Encoding**
  - Count the number of times a single color appears in a row
  - Split data into chunks based on end of color, line, or count capacity

<table>
<thead>
<tr>
<th>Count</th>
<th>4-bit Color Lookup Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x01</td>
<td>0 0 0 1</td>
</tr>
<tr>
<td>0x03</td>
<td>0 0 0 0</td>
</tr>
<tr>
<td>0x01</td>
<td>0 0 0 1</td>
</tr>
</tbody>
</table>
Goal #1: View the entire map

- Map is too large to view in one piece
- Scrolling with the keyboard is necessary
- Requires keeping track of current view
- Alternate line buffers and account for sprites

```
get_background_position_from_device(void)
write_background_position(background_window_pos *background, char axis)
```

Register 0: Scroll Display

<table>
<thead>
<tr>
<th>(MSB)</th>
<th>31:25</th>
<th>24</th>
<th>23:17</th>
<th>16</th>
<th>15:12</th>
<th>11:0</th>
</tr>
</thead>
<tbody>
<tr>
<td>padding</td>
<td></td>
<td></td>
<td>padding</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>axis</td>
<td>0 = x</td>
<td></td>
<td>jump</td>
<td>0 = smooth</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1 = y</td>
<td></td>
<td>1 = jump</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>padding</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>value</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Animating Trains

Register 2: Move Sprite to Location

<table>
<thead>
<tr>
<th>(MSB)</th>
<th>31:24</th>
<th>23</th>
<th>22</th>
<th>21:11</th>
<th>10:0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Sprite #</td>
<td>jump</td>
<td>padding</td>
<td>Y Position</td>
<td>X position</td>
</tr>
</tbody>
</table>

- Shifted gradually by hardware
  - Unless jump = 1
- Sprite number used to specify which train
Add-On: Web View

- View output without looking at screen
- Allows for remote development
- Data from the same source of the board
- Controls the screen and inputs remotely
Goal #2: Animating Trains

The MTA API is not very good.

- Only gives expected arrival times which are often inaccurate
- We must guess where the train is based on finding local minima arrival times
- The station IDs are not easy to attribute to certain lines
Goal #3: Update One Line With Live Data

- Need to query live MTA subway data
- Python client/server to query MTA API and wait for line queries
- C code polls python client/server at frequent intervals

```c
while(train_interface_thread_running)
{
    fetch_trains(latest_line_data, line_number);

    if(NULL != latest_line_data->trains &&
        latest_line_data->number_of_trains != 0)
    {
        populate_active_line_with_trains(latest_line_data->
            trains, latest_line_data->number_of_trains);
    }

    free_trains(latest_line_data);
    usleep(INTERFACE_REFRESH_INTERVAL);
}
```
Drawing Map and Train Pathing

- Trains must follow their respective line
- Station locations must be known to place trains in the right segments

How?

- GIMP!
- And some C to record coordinates of certain colors
Goal #4: Switching Between Lines

- Clears the sprites from the screen
- Gets new API data
- Creates new sprites with updated graphics
Demo Time!