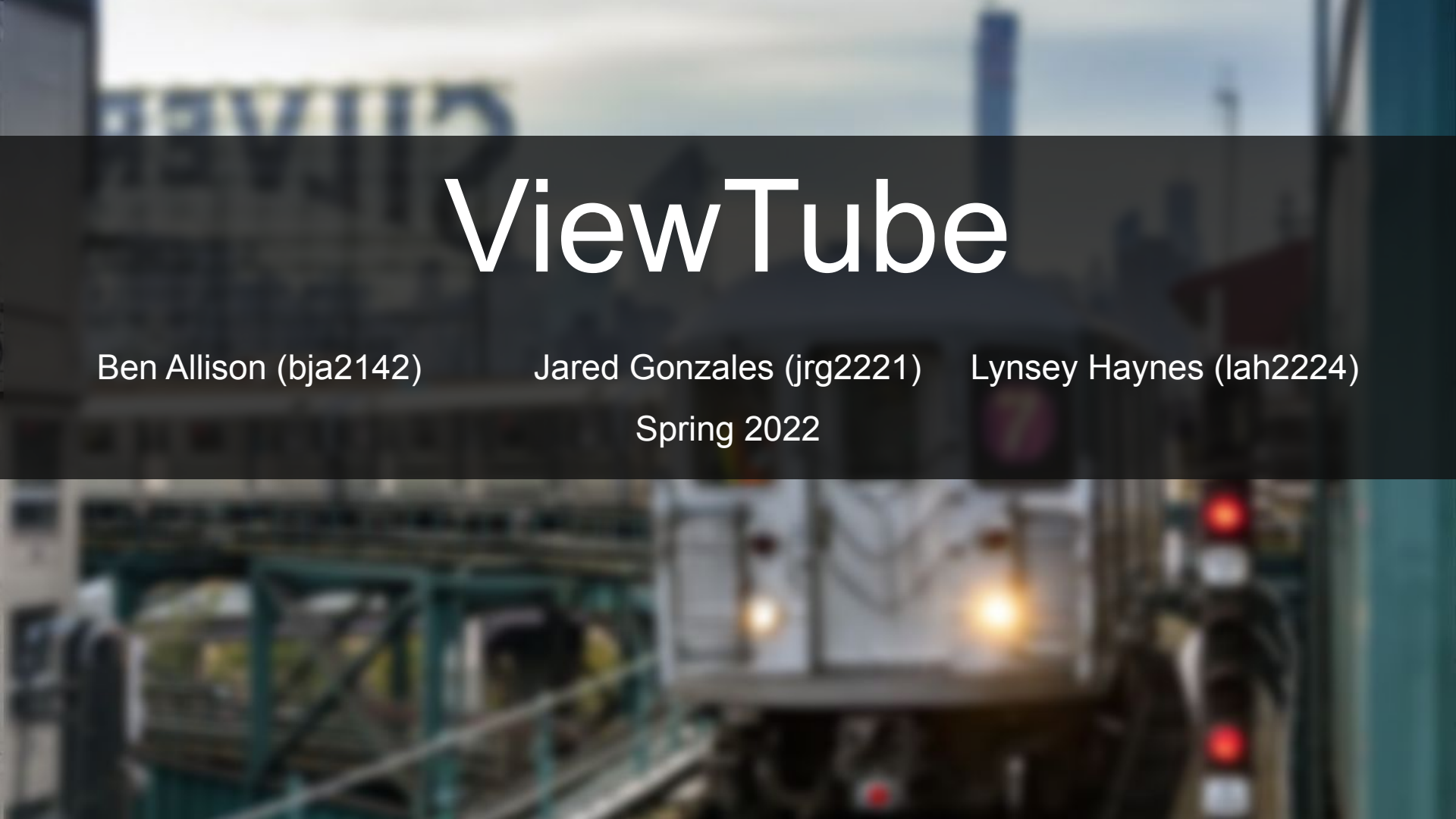
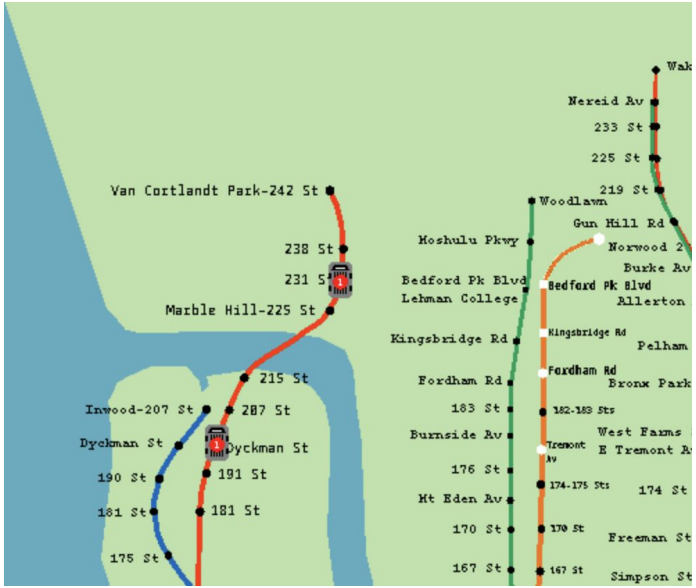


ViewTube

Ben Allison (bja2142) Jared Gonzales (jrg2221) Lynsey Haynes (lah2224)

Spring 2022

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

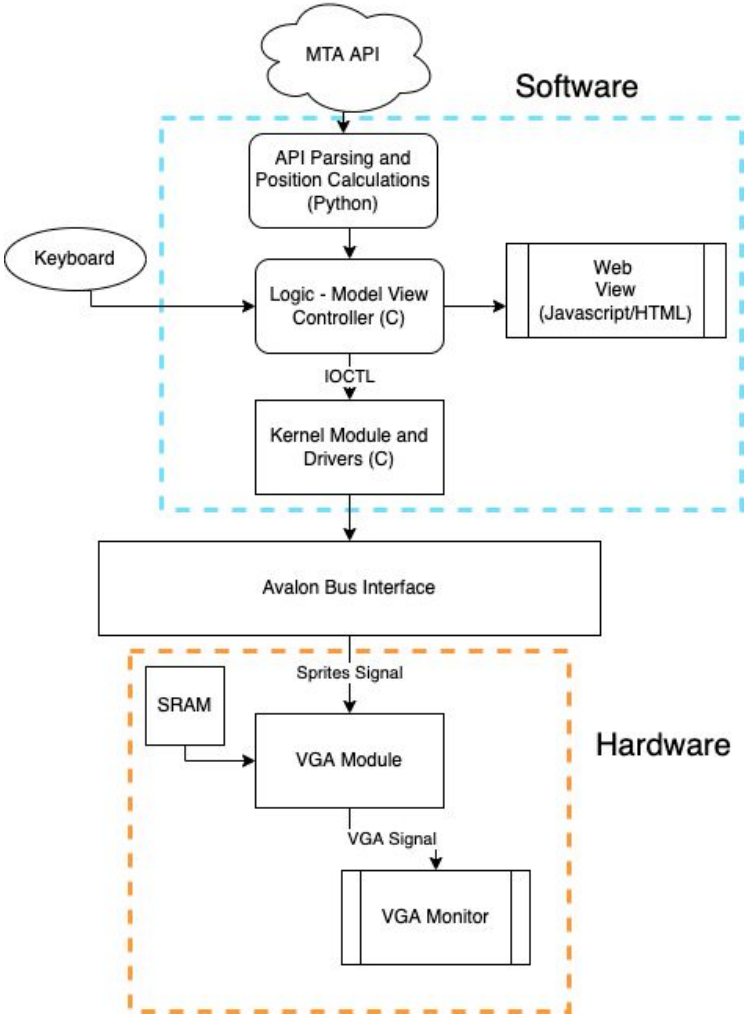
Implementation

Software

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

Hardware

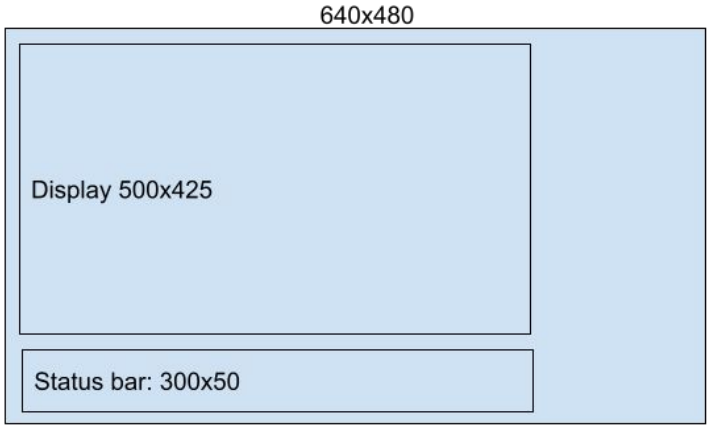
- Draws to the screen
- Updates sprite positions



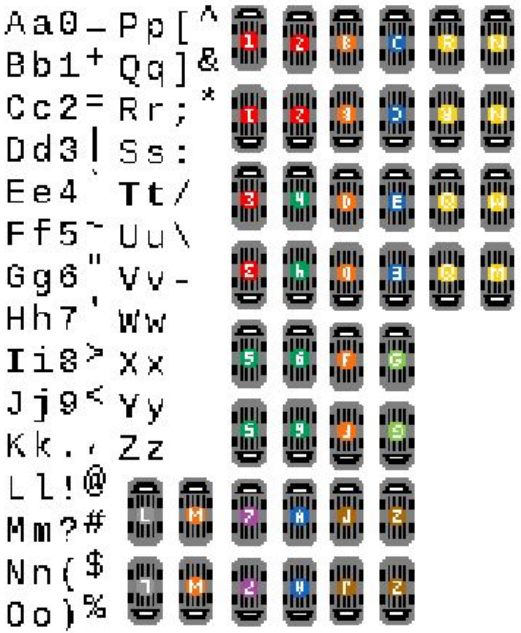
Graphics

Two layers of graphics:

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



Sprites



Register 1: Update Sprite in Table

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

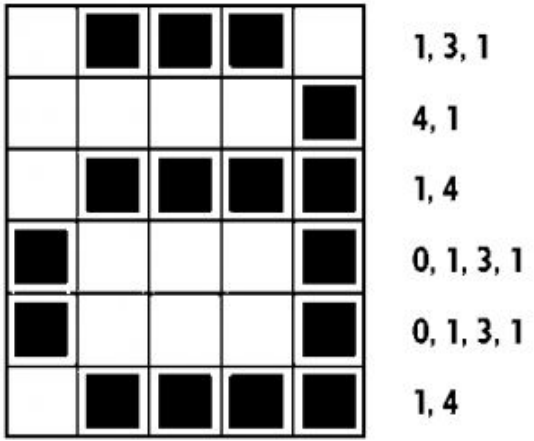
(MSB)	31:24	23	22:21	20:16	15:13	12:8	7:5	4:0
	Table Offset	type	padding	New Sprite #	padding	width	padding	height

Type 1: 1-bit color for letters

(MSB)	31:24	23	22:16	15:13	11:8	7:4	3:0
	Table Offset	type	New Sprite #	padding	red	green	blue

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

Count	4-bit Color Lookup Code			
0x01	0	0	0	1
0x03	0	0	0	0
0x01	0	0	0	1

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

(MSB)	31:25	24	23:17	16	15:12	11:0
	padding	axis 0 = x 1 = y	padding	jump 0 = smooth 1 = jump	padding	value

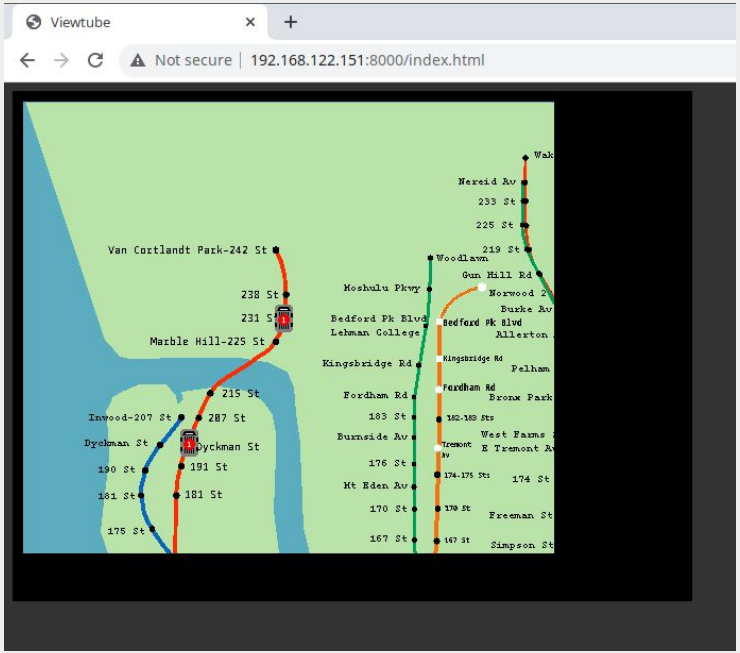
Animating Trains

Register 2: Move Sprite to Location

(MSB)	31:24	23	22	21:11	10:0
	Sprite #	jump	padding	Y Position	X position

- 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

```
Current Time:01:32:58
==================NORTHBOUND TRAINS=====

Pelham  01:45:43
Buhre   01:43:13
Middle  01:42:13
Westch  01:40:43
Zerega  01:39:43
Castle  01:38:43
Parkch  01:36:43
St Law  01:35:13
Morris  01:34:13
Elder   01:33:13  TRAIN! STOPPED
Whitlo  01:42:26
Hunts   01:38:26
Longwo  01:36:26
E 149   01:34:56
E 143   01:33:26
TRAIN! Arrives in 8 seconds
Cypres  01:41:08
Brook   01:40:08
3 Av -  01:36:08
TRAIN! Arrives in 170 seconds
125 St  01:47:58
116 St  01:46:28
110 St  01:44:58
103 St  01:43:58
96 St   01:42:28
86 St   01:40:28
77 St   01:38:58
68 St   01:37:28
Lexing  01:35:28
51 St   01:33:58
TRAIN! Arrives in 40 seconds
Grand   01:47:00
33 St   01:45:00
```

Goal #3: Update One Line With Live Data

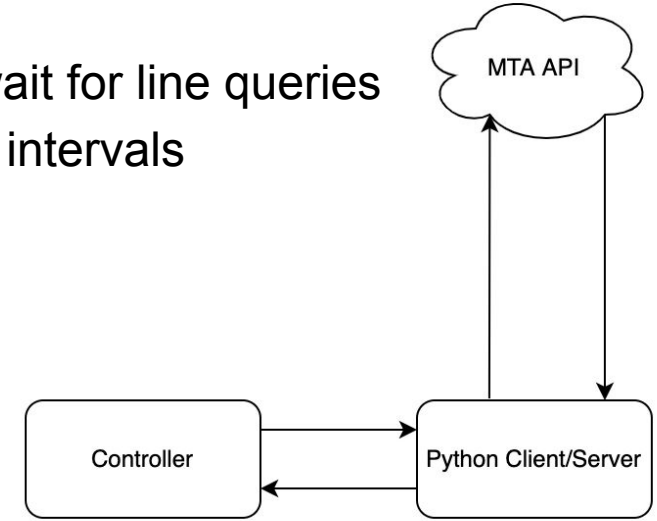
- Need to query live MTA subway data
- Python client/server to query MTA API and wait for line queries
- C code polls python client/server at frequent intervals

```
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!