Correct, The direction coordinates must remain the same for any track, no effect on track layout.

I will try to explain, there are 32 car graphics, each graphic points in a different direction.

0 degrees top of screen.
90 degrees right side of screen.
180 degrees bottom of screen.
270 degrees left side of screen.
360 degrees top of screen.

Each car graphic has a direction coordinate, so 32 directions. These bytes are stored in the ROM from location $00 to $1F. Now we need one more, a code for stop which is at location $20.

The output of the steering wheel is used as an address to the ROM that selects the car graphic and direction code to use. Think of the screen as a piece of graph paper and remember we can only move in the horizontal and vertical directions.

The vertical and horizontal stop codes are contained in one byte.
Bits 0 → 3 equal horizontal stop code
Bits 4 → 7 equal vertical stop code.

The vertical motion is controlled by a 12 bit counter (max count 4096 or ($000 to $FFF)). At the start of a game, bits 4 → 11 are set to $DFx and the stop code of $7 (bits 0 → 3 of the direction code) is set to bits 0 → 3 of the counter for a value $DF7 this is the stop code for vertical motion. 4096 minus 3575 ($DF7) equals 521 which is the total number of vertical lines for the game.

Every 521 lines will be the same vertical position on the screen. By changing the 4 least significant bits we can control the vertical movement. With a range of $0 to $F and $7 equaling stop.

4096 minus 3568 ($DF0) equals 528 lines
4096 minus 3583 ($DFF) equals 513 lines
4096 minus 3575 ($DF7) equals 521 lines Stopped

Possible vertical moves
521 minus 528 equals -7 or 7 lines down
521 minus 513 equals 8 or 8 lines up

Horizontal works the same way except $0 is the stop code (Note bit 3 gets inverted so counter sees an $8 as the stop code).

ROM data – direction codes

```
0000:    b0 bf af ae ad 9d 8d 8c 7c 6c 6d 5d 4d 4e 4f 3f
0010:    30 31 41 42 43 53 63 64 74 84 83 93 a3 a2 a1 b1
0020:    70
```

Let's look at the first car graphic in the ROM, it's direction is zero degrees (pointing to the top of the screen). ROM memory location $0000 contains the direction code $B0

4096 minus 3579 ($DFB) equals 517
521 minus 517 equals 4 or 4 lines up

With horizontal code being stop ($0) no movement left or right.

Car moves 4 lines up the screen.

Code	Movements	
B0	4 up 0 right	
BF	4 up 1 right	
AF	3 up 1 right	
AE	3 up 2 right	
AD	3 up 3 right	45 degrees
9D	2 up 3 right	
8D	1 up 3 right	
8C	1up 4 right	
7C	0 up 4 right	90 degrees
6C	1down 4 right	
6D	1 down 3 right	
5D	2 down 3 right	
4D	3 down 3 right	135 degrees
4E	3 down 2 right	
4F	3 down 1 right	
3F	4 down 1 right	
30	4 down 0 right	180 degrees
31	4 down 1 left	
41	3 down 1 left	
42	3 down 2 left	
43	3 down 3 left	225 degrees
53	2 down 3 left	
63	1 down 3 left	
64	1 down 4 left	
74	0 up 4 left	270 degrees
84	1 up 4 left	
83	1 up 3 left	
93	2 up 3 left	
A3	3 up 3 left	315 degrees
A2	3 up 2 left	
A1	3 up 1 left	
B1	4 up 1 left	
70	Stop	

With this information we have the direction code matched to the car graphics. But how does the car rotate in the same spot if there is no gas signal ?

Simple, Remember the speed pulse, if no gas it is a low signal as gas increases it starts to pulse high the more gas the longer the high pulse until it stays high which is full speed.

If the pulse is low the direction code is held stopped by forcing the ROM address to $20 which is the stop code data $70 so the car maintains it's position. When a speed pulse shows up on a frame the direction code is loaded and the car moves for that frame.

For 30 frames per second if one frame has a high speed pulse and car direction is 0 degrees the car moves 4 vertical lines up. Relates to 4 up per second. But if 10 of those frames had high speed pulses then the car would move a total 40 vertical lines up per second. 10 frames at 4 up each.

Hope this helps.

Tim