



Moto (A)GP(X) 2023



Descrizione del progetto

Introduzione

Vi siete mai chiesti come fanno giochi tipo “Pitstop II” a disegnare ed animare quella bella strada in maniera così veloce e fluida? Io sì e ho voluto formulare una mia versione. In questo documento spiegherò la mia idea ed implementazione, che ho visto successivamente ricalcare molte dell’idee del gioco succitato.

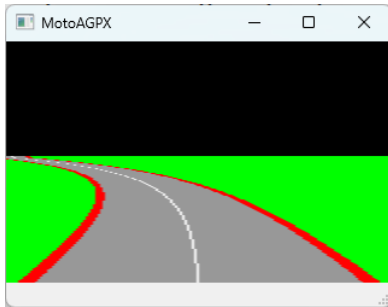
L’idea è quella di memorizzare una sequenza di schermate rappresentanti le varie strade (da quella con la curvatura più a sinistra, fino a quella più a destra). Siamo in modalità testo multi colore. Le varie schermate vengono precalcolate da un programma scritto in C++, che esporterà la sequenza di byte da inserire nel programma BASIC.

Per evitare il crescere eccessivo del numero di strade, i cordoli e la striscia centrale vengono tracciati in maniera continua (senza “zebratura”) e poi si affida ad un raster interrupt l’onere di suddividerle e animarle.

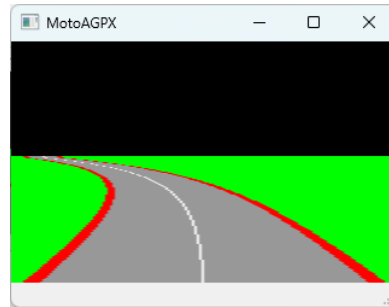
Inizialmente memorizzavo tutte le strade in formato RLE, ma mi sono accorto che il frame rate scendeva spesso sotto i 50 fps (nonostante le routine in assembly scritte a mano), ma poi ho trovato una soluzione che vedrete nelle slide seguenti.

Quindi un programma BASIC compilato ma, più che mai in questo caso, abbiamo anche un bel po’ di codice scritto in assembly perché i tempi sono troppo stretti (l’assembly non è morto, nel C64 è un MUST, tuttavia il BASIC compilato è un discreto “direttore d’orchestra”)!

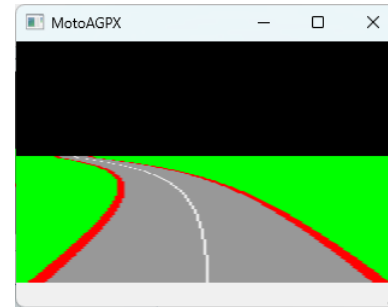
Il numero di strade che ho deciso di creare sono 29, così suddivise: 14 curvature a sinistra, 1 centrale e 14 curvature a destra (in quest'ordine). Il numero 14 corrisponde alla costante NUM_ROADS nel BASIC. Ogni strada ha quindi un indice ('roadlx') da 0 a 28 (in generale sono $2 * \text{NUM_ROADS} + 1$). La strada comincia alla riga #12 dello schermo e termina alla #24 (zero-based).



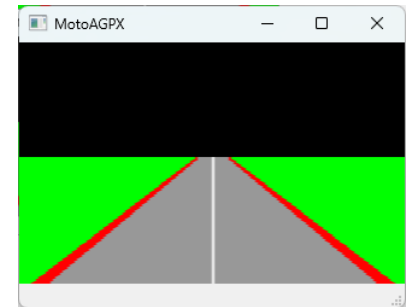
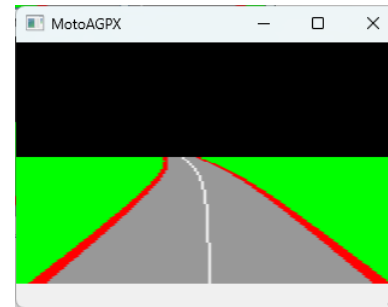
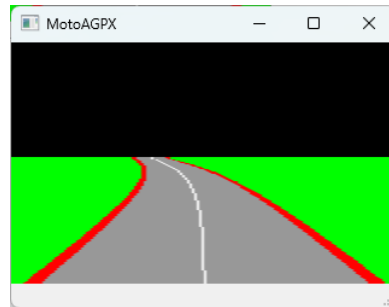
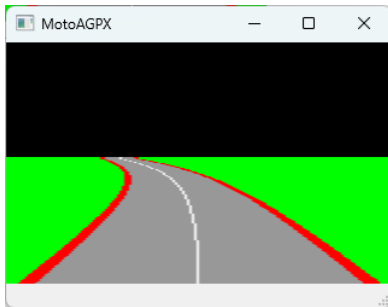
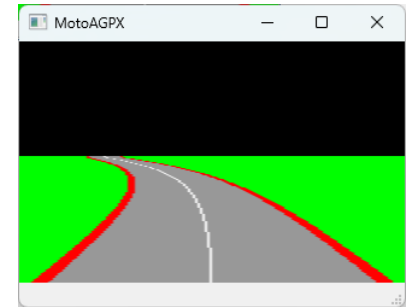
roadlx = 0



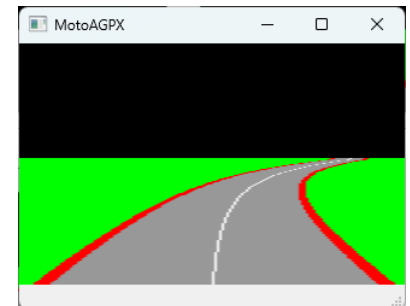
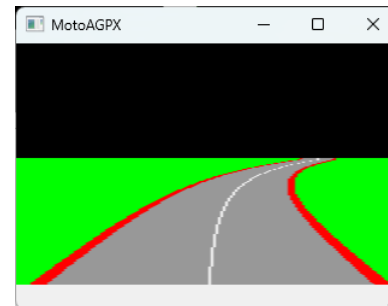
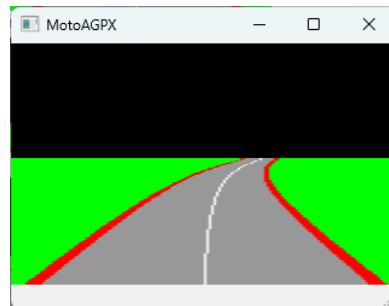
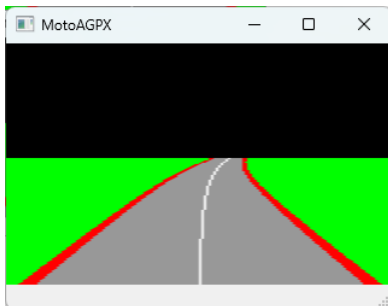
roadlx = 1



roadlx = 2

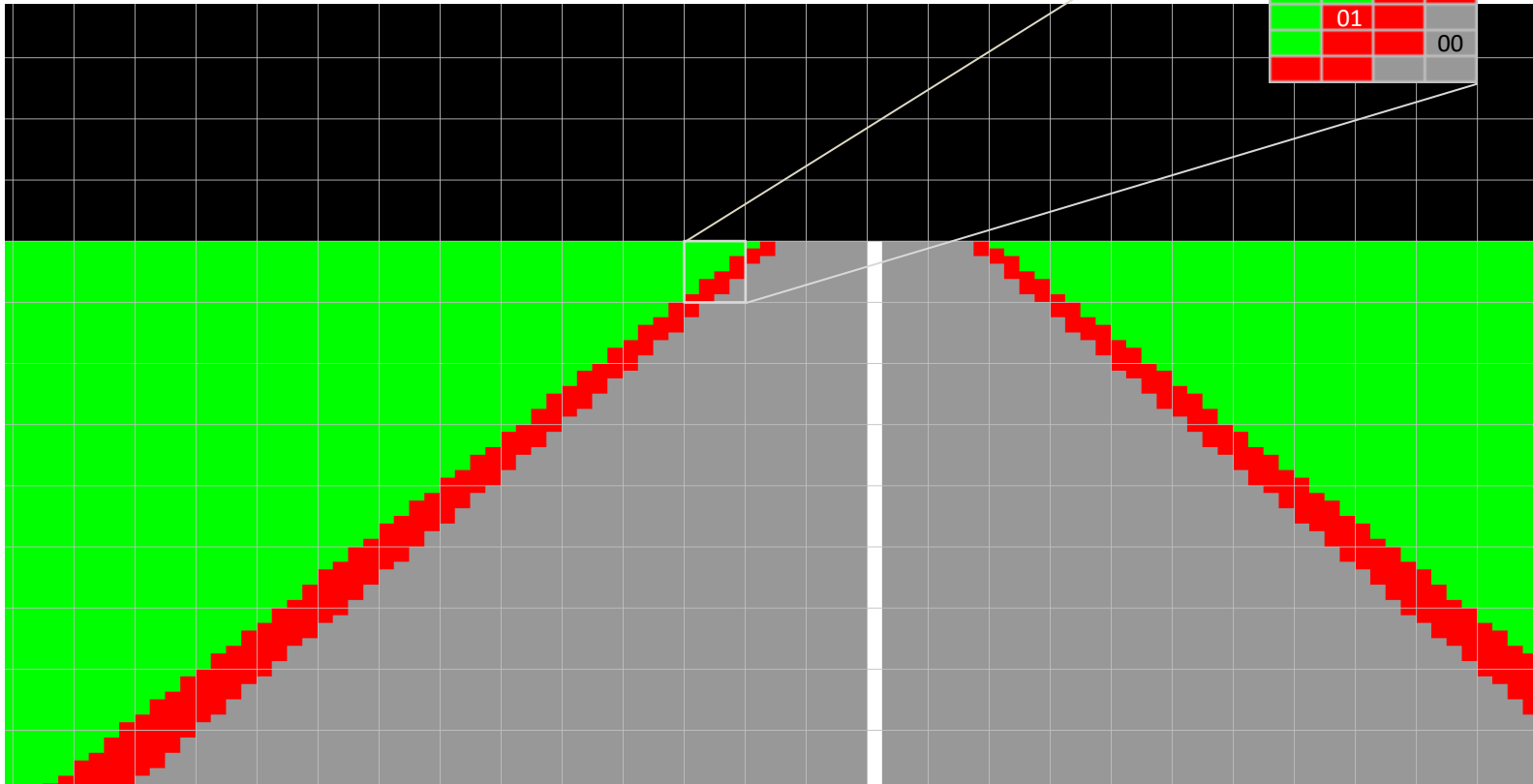


roadlx = 14 (NUM_ROADS)



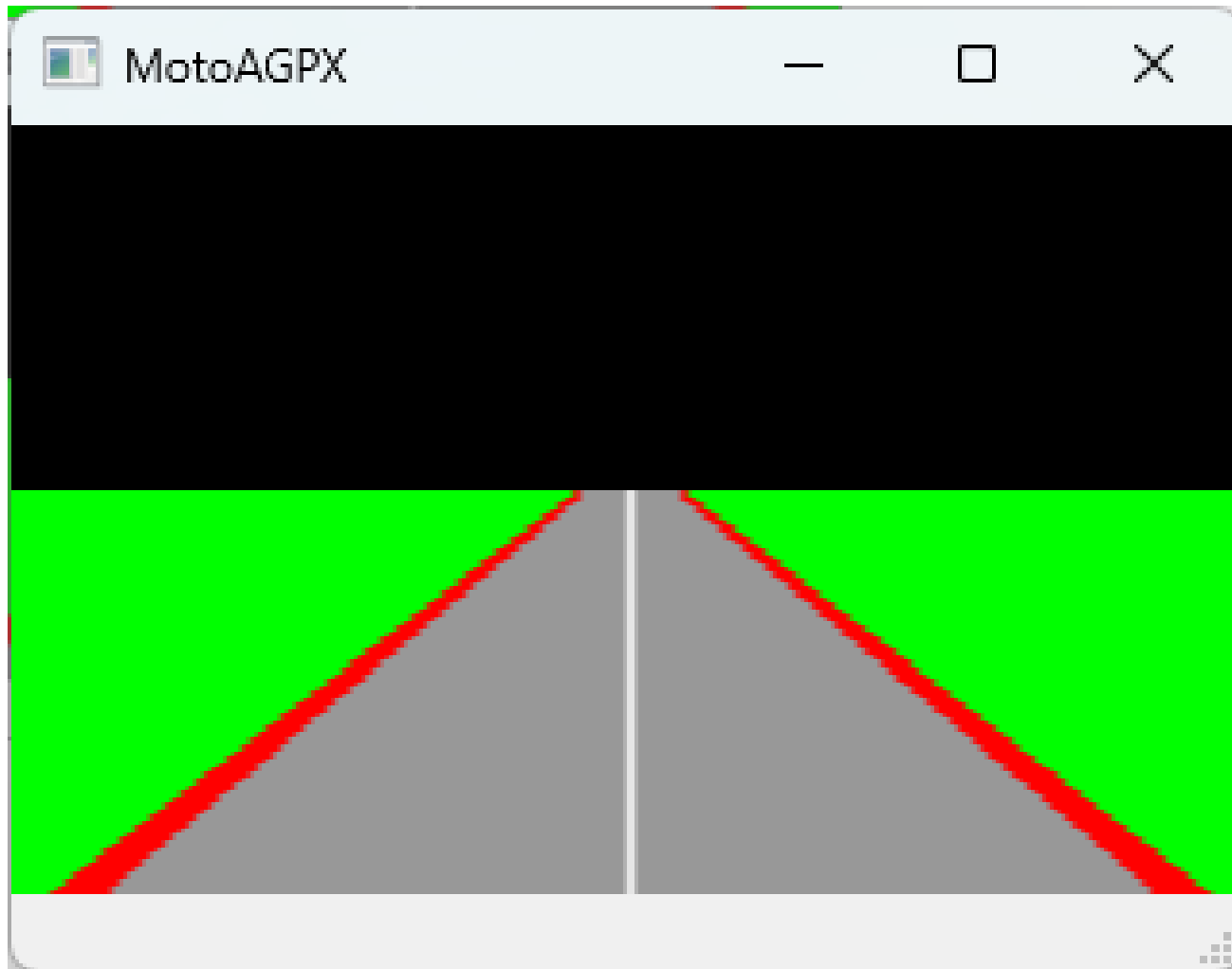
roadlx = 28 ($2 * \text{NUM_ROADS}$)

Il programma C++ divide ogni fotogramma in blocchetti da 4 x 8 pixel e genera i byte del relativo carattere ridefinito (00 = grigio, 01 = rosso, 10 = bianco, 11 = verde). I caratteri duplicati vengono ignorati. L'elenco di tutti i caratteri ottenuti, cioè di tutte le strade senza duplicati (sono 1101), si trova nella zona iniziale dei nostri dati.

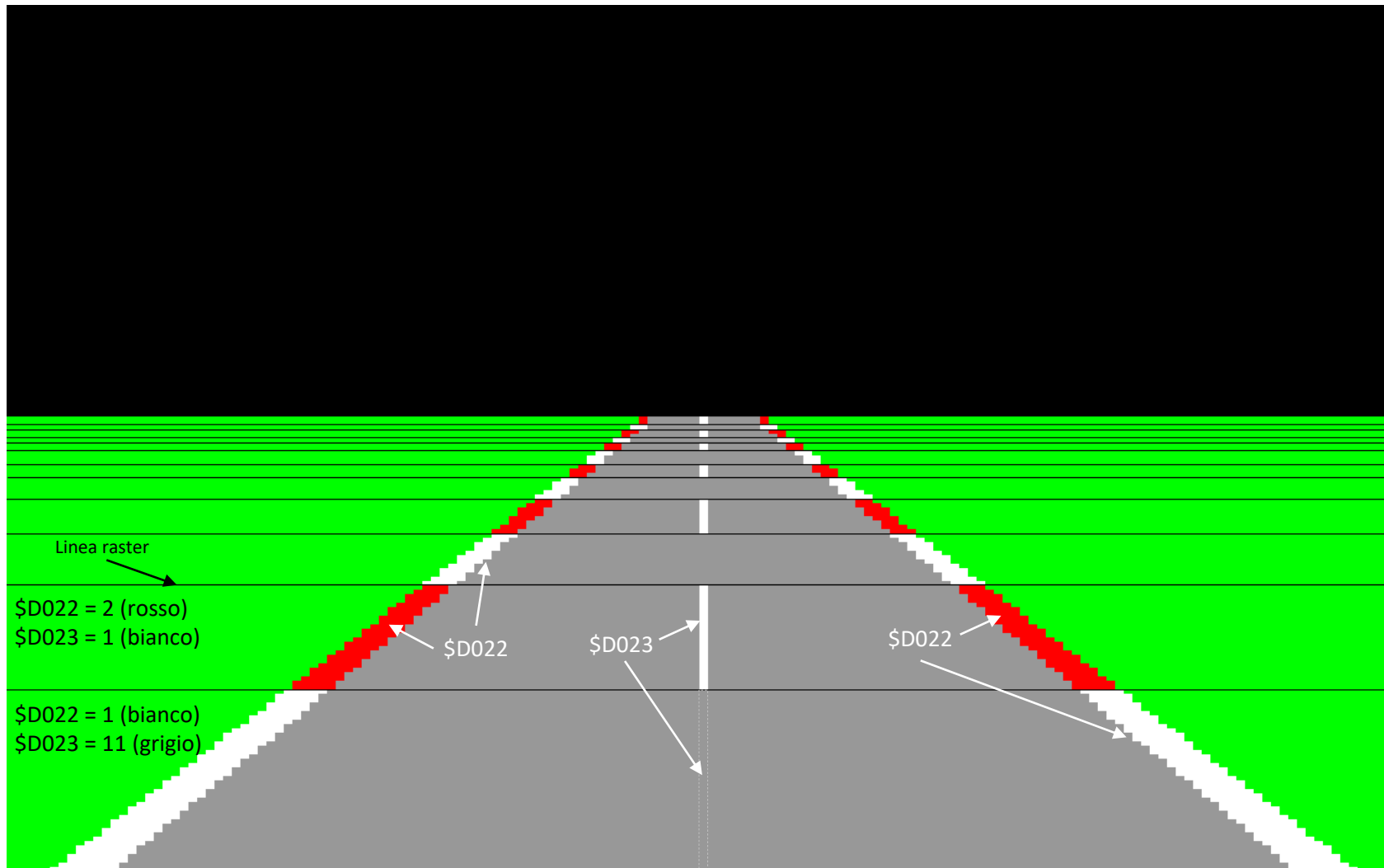


Un carattere è fatto da 8 byte, quindi con un intero senza segno a 64 bit (uint64_t) si può memorizzare un carattere. In C++, l'elenco completo viene memorizzato in una tabella hash, dove la chiave è il carattere (a 64 bit) ed il valore è l'indice del carattere nel nostro array finale. In questo modo, in maniera molto veloce, durante la scansione dello schermo si può determinare se un carattere è duplicato (chiave esistente).

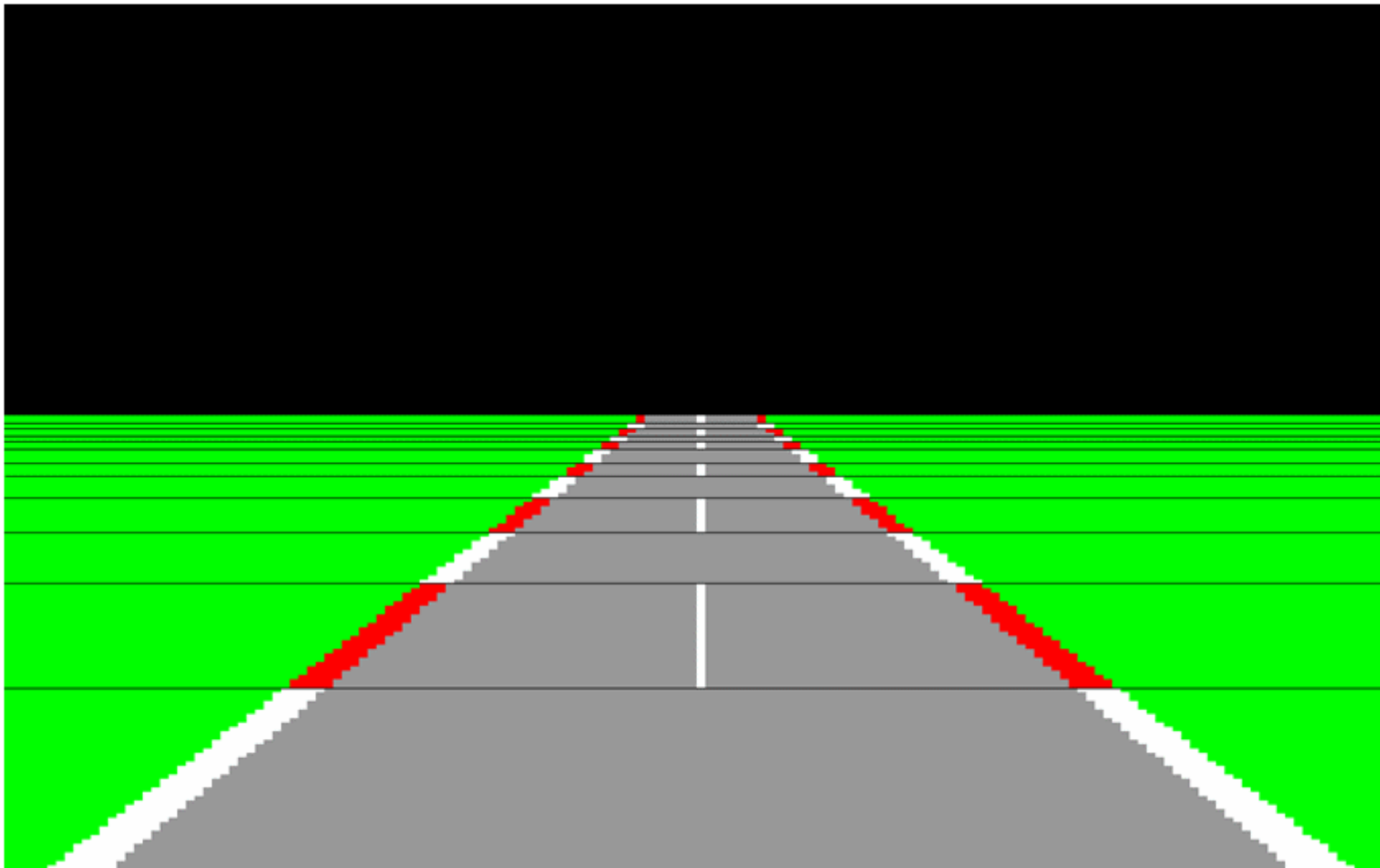
I cordoli laterali (in rosso) e la striscia centrale bianca sono continui. I cordoli utilizzano il colore multi 1 (\$D022), mentre la striscia centrale il multi 2 (\$D023). Il colore di foreground (\$D800 - \$DBE7) è verde (per l'erba), mentre la strada è trasparente (\$D021).



Una routine di raster cambia \$D022 e \$D023 in modo da colorare i cordoli e creare interruzioni nella linea centrale (che poi, sempre di cambio di colore si tratta). Lo schermo è stato suddiviso in 12 strisce (costante ROAD_STRIPS nel codice basic). Renderla, però, stabile non è affatto una cosa semplice!



Queste 12 linee raster vengono shiftate in basso (sempre tenendo conto della prospettiva), per un totale di 32 insiemi (costante STRIP_STEPS). Il passo con cui si muovono è studiato affinché al 32-esimo step, i cordoli ritornino come al passo 0: in questo modo si avrà un effetto di movimento continuo. Avremo quindi una tabella di $12 * 32$ byte (all'inizio dei nostri dati), che rappresentano tutte le possibili posizioni delle linee raster e ci serviranno appunto per creare l'effetto di scorrimento di tutte le strade. L'immagine seguente mostra soltanto 2 delle 32 posizioni possibili:



Con il C64 bisogna andarci piano con l'overdraw, ovvero il tracciamento della stessa area più volte, soprattutto quando essa non è cambiata. Per questioni di prestazioni è meglio che, quando la strada deve curvare, evitiamo di tracciare cose che non cambiano (tipo molte zone dell'erba o della strada). Per realizzare ciò, si memorizzano soltanto le differenze tra la strada di indice i e la sua successiva $i + 1$ (e anche con la precedente $i - 1$). Ad esempio per memorizzare le differenze con la strada successiva, ho utilizzato lo schema seguente:

06	; 6 caratteri differenti
E8 01	; offset da cui disegnare ($\$01E8 = 488 = 12 * 40 + 8$, ovvero dall'8° carattere della riga 12)
02 01 04 05 08 03	; i 6 caratteri che vanno scritti
03	; 3 caratteri differenti
F5 01	; offset da cui disegnare ($\$01F5 = 501 = 12 * 40 + 21$)
05 0B 08	; i 3 caratteri che vanno scritti
...	
00	; fine della schermata

Uno 0 come numero di caratteri differenti, indicherà la fine delle differenze.

Si possono trovare schemi più compatti, ma questo è molto veloce nella fase di decompressione.

Si noti che con questo trucco, ho lo svantaggio che per passare dalla strada i alla strada $i + 2$, dovrò necessariamente prima passare per la $i + 1$, poiché sono in grado di evolvere il quadro attuale solo al successivo o al precedente.

Per mettere in atto il trucco precedente, è necessario memorizzare almeno il primo quadro (strada dritta, roadIx = NUM_ROADS) per intero, poiché ovviamente inizialmente lo schermo è vuoto. Per far ciò ho compresso la schermata usando il formato RLE (Run Length Encode) che memorizza i caratteri come segue:

09	; 9 caratteri uguali
01	; carattere da ripetere (da 0 a 255)
8A	; 1 carattere solo (8° bit acceso), il cui valore è 0A (i 7 bit meno significativi)
03	; 3 caratteri uguali
61	; carattere da ripetere (da 0 a 255)
...	
FF	; indica la fine dell'immagine

L'esempio si decompone in: **01 01 01 01 01 01 01 01 01 01 0A 61 61 61** (6 byte anziché 13).

Fortunatamente il numero di caratteri ridefiniti per una singola schermata è minore di 128, quindi posso eliminare il byte di ripetizione quando ho un singolo carattere, sfruttando l'8° bit come flag. Avremo quindi **8A** anziché **01 0A**, nel nostro esempio.

Gli ingredienti per disegnare la strada ci sono tutti! Questi dati vengono esportati dal programma C++ in un unico blocco (vedi etichetta 'road:' nel sorgente BASIC) e ammontano alla bellezza di 25714 byte, che non sono pochi, ma il compilatore riesce a gestirli. Il formato esatto dei dati è il seguente:

Offset	Descrizione	Dimensione (in byte)
0	Tabella posizioni delle rasterline (12 posizioni per 32 variazioni)	$(ROAD_STRIPS * STRIP_STEPS) = (12 * 32) = 384$
384	Tabella dei caratteri per tutte le varianti delle strade (1101 caratteri)	$CHARS_TABLE_SIZE = 1101 * 8 = 8808$
9192	Tabella degli offset delle strade. Contiene un indirizzo (2 byte) per ognuno delle possibili strade, che punta alla zona di memoria che contiene la descrizione della strada (quali caratteri usa dei 1101, caratteri della strada precedente e successiva, eventuale compressione RLE). Ad esempio per la strada 0, sarà 9250.	$(2 * NUM_ROADS + 1) * 2 = (2 * 14 + 1) * 2 = 58$
9250	Numero di caratteri ridefiniti (nc) della strada di indice 0	1
9251	Tabella di offset (nella tabella dei 1101 caratteri) degli nc caratteri	$nc * 2$
$9251 + nc * 2$	Dimensione in byte (drle) del frame in formato RLE (se manca, vale 0)	1
$9252 + nc * 2$	Eventuali dati del frame RLE	$drle (< 256)$
$9252 + nc * 2 + drle$	Dimensione (dprev) del "frame precedente"	1
$9253 + nc * 2 + drle$	Dati del "frame precedente"	$dprev (< 256)$
$9253 + nc * 2 + drle + dprev$	Dimensione (dnext) del "frame successivo"	1
$9254 + nc * 2 + drle + dprev$	Dati del "frame successivo"	$dnext (< 256)$
$9254 + nc * 2 + drle + dprev + dnext$	Numero di caratteri ridefiniti (nc) della strada di indice 1	1
e così via per tutte le altre $(2 * NUM_ROADS)$ varianti della strada		

Sebbene la tabella globale dei caratteri (offset 384) contenga 1101 caratteri, solo quelli necessari (nc) alla particolare strada vengono copiati a run-time nella memoria caratteri. Essi sono stati ovviamente ri-indicizzati a partire da 0. Infine, la tabella su esposta viene esportata considerando come offset di partenza 0, ma nella memoria del C64 corrisponderà ad un indirizzo ben preciso assegnato dal compilatore. Per tale ragione, la routine 'relocate Data', aggiusta gli offset necessari (ad esempio la tabella degli offset a partire da 9192) in modo da trasformali in indirizzi fisici ed evitare di dover fare la somma a run-time ogni volta.

Il tracciamento della strada richiede tempo e provoca inevitabilmente flickering e artefatti vari, considerando anche che dobbiamo ogni volta ridefinire i caratteri. Per evitare questo, si usano due pagine video: una visibile e una nascosta (**double buffering**).

“VIC-II Calculator” alla mano, ho deciso di usare il 3° banco come segue:

La pagina 0, ha memoria caratteri in **\$E800-\$EFFF** e memoria video in **\$F000-\$F3FF**

La pagina 1, ha memoria caratteri in **\$F800-\$FFFF** e memoria video in **\$F400-\$F7FF**

Le 4 aree non si sovrappongono e in totale occupano 6Kb (gli ultimi 6 dei 64Kb), così lasciamo la zona bassa libera per il BASIC.

Quando la pagina 0 è visibile (tramite registro \$D018), posso tranquillamente scrivere i miei caratteri in \$F800 e disegnare la mia strada in \$F400. Appena ho finito, la pagina 1 diventa visibile (sempre tramite registro \$D018) e la 0 diventa nascosta. A questo punto scriverò nella 0 e la cosa si ripete scambiando continuamente il ruolo della pagina 0 e la 1 (da visibile a nascosta).

In pratica si scrive sempre nella nascosta e appena pronta la visualizzo fulmineamente.

L'algoritmo per tracciare la strada di indice i , il cui precedente era ad esempio $i - 1$, effettua i seguenti passaggi:

1. Accede la tabella degli offset delle strade (9192) per ottenere l'indirizzo di memoria che descrive la strada i -esima;
2. Da qui copia gli nc caratteri dalla tabella globale a quella dei caratteri della pagina nascosta (a partire dal carattere 0);
3. Se è stato chiesto di tracciare la strada usando il formato RLE, lo esegue;
4. Altrimenti, disegna solo le differenze prelevando le informazioni dalla tabella relativa alla strada precedente ($i - 1$);

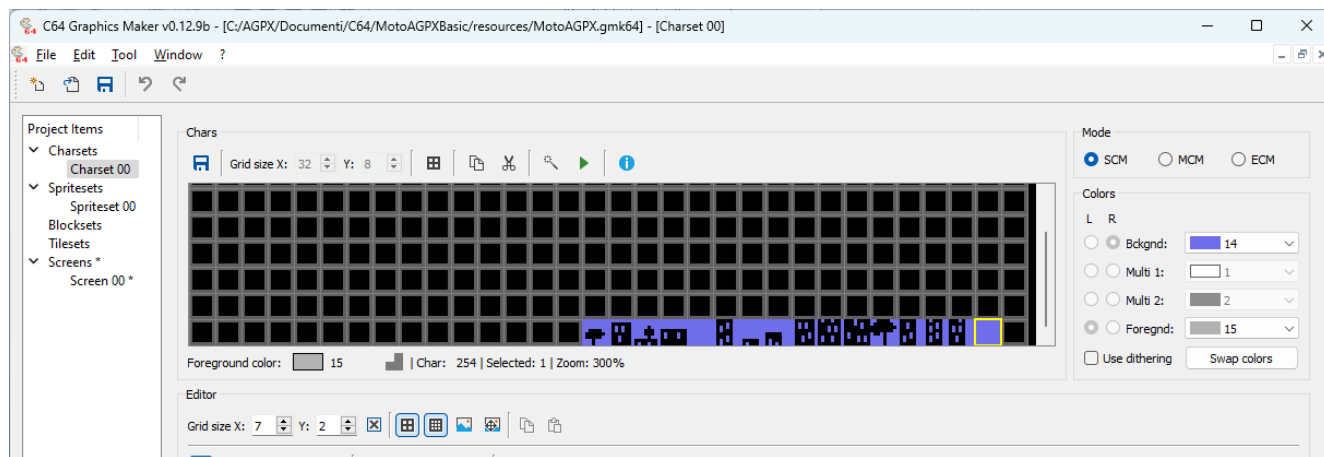
Ogni volta che disegniamo in una pagina, annotiamo l'indice della strada perché ci servirà per il prossimo tracciamento. Tali indici vengono memorizzati nelle variabili di nome `roadlxPage0` e `roadlxPage1`.

Per quanto riguarda il circuito, esso è memorizzato in un'array che contiene due informazioni per voce: la distanza dalla partenza e l'indice di strada. Un'estratto della mappa di prova è fatta come segue:

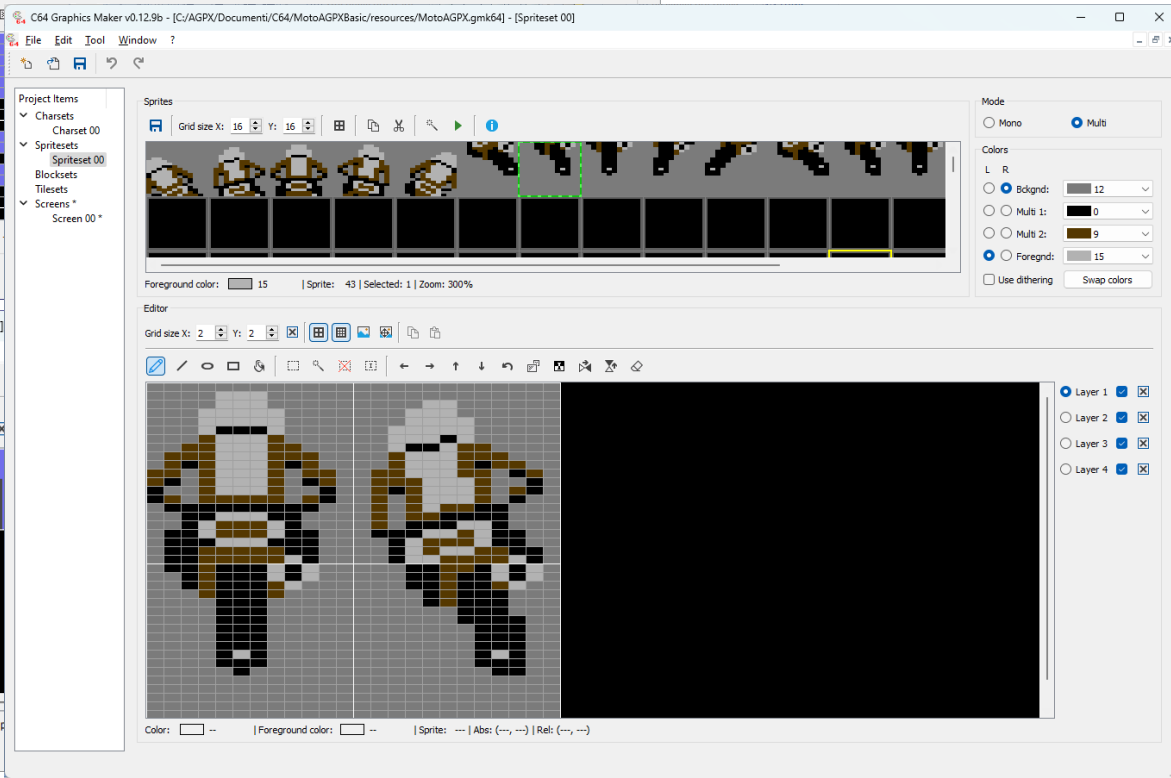
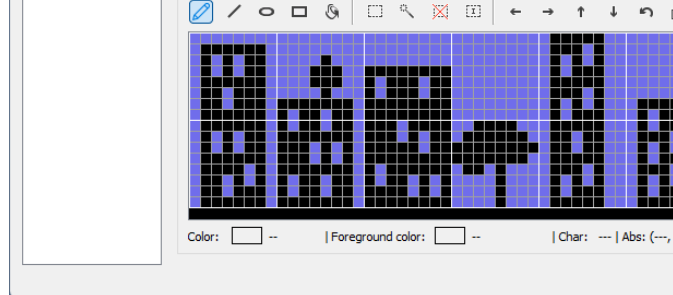
Distanza in cm dalla partenza (dcm)	Indice strada (roadIx)	Descrizione
3000	NUM_ROADS	30 m di rettilineo
9000	$\text{NUM_ROADS} + (\text{NUM_ROADS} / 2)$	Curva leggermente a destra in 60 m
15000	NUM_ROADS	Ritorna dritta in 60 m
24000	NUM_ROADS	Rettilineo di 90 m
30000	$\text{NUM_ROADS} + (\text{NUM_ROADS} / 2)$	Curva leggermente a destra in 60 m
36000	$\text{NUM_ROADS} - (2 * \text{NUM_ROADS} / 3)$	Curva brusca a sinistra in 60 m
42000	NUM_ROADS	Ritorna dritto in 60 m
45000	NUM_ROADS	Rettilineo di 30 m
48000	$\text{NUM_ROADS} * 2$	Super curva a destra in 30 m
51000	NUM_ROADS	Torna dritto in 30 m

Essenzialmente il circuito è fatto da “segmenti” (tipo MapSegment) di strada, disposti uno dopo l'altro. Si noti che giocando sull'indice di strada e sulla lunghezza del segmento, si possono creare curve più o meno difficili.

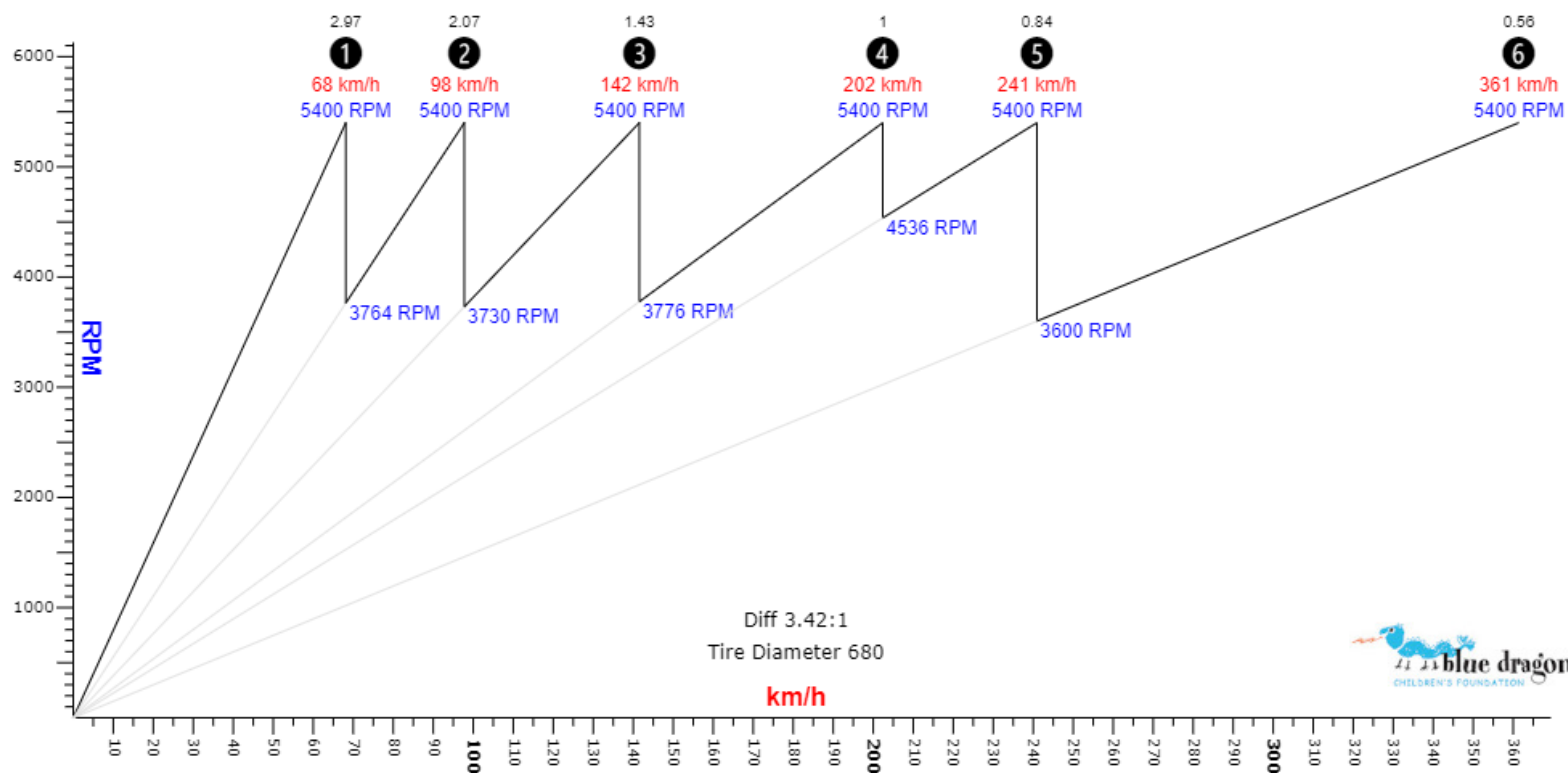
Sprite, caratteri e fondali sono realizzati tramite il solito C64 Graphics Maker (ok, la moto fa schifo, a voi fare di meglio!):



I caratteri partono dall'indice 254 (spazio vuoto) fino al 128, poiché quelli che vanno da 0 a 127 sono riservati per i caratteri della strada. Anche le lettere e i numeri vengono copiati (una volta sola) a run-time (per risparmiare spazio) dalla ROM caratteri.



Il suono della motocicletta è stato realizzato con l'aiuto di un sito (<https://www.blocklayer.com/rpm-gear>) che mostra un grafico del numero di giri in funzione della velocità in Km/h e del cambio marcia (mi sono spinto fino alla 5ª):



La frequenza emessa (forma d'onda quadra o dente di sega) è proporzionale al numero di giri del motore. In questo modo si simula anche il cambio di marcia durante l'accelerazione.

Per finire un elenco di possibili miglioramenti:

- Aggiungere un contatore iniziale che dia il via;
- Creare più circuiti;
- Aggiungere la visualizzazione della mappa;
- Aggiungere una linea di partenza (con degli sprite);
- Migliorare gli sprites;
- Migliorare la grafica di fondo;
- Macchie d'olio sull'asfalto (sprite) con cadute;
- Cartelli a bordo strada (sprite);
- Aggiungere degli avversari;
- Aggiungere il carburante ed il pit stop;

That's All Folks!

AGPX