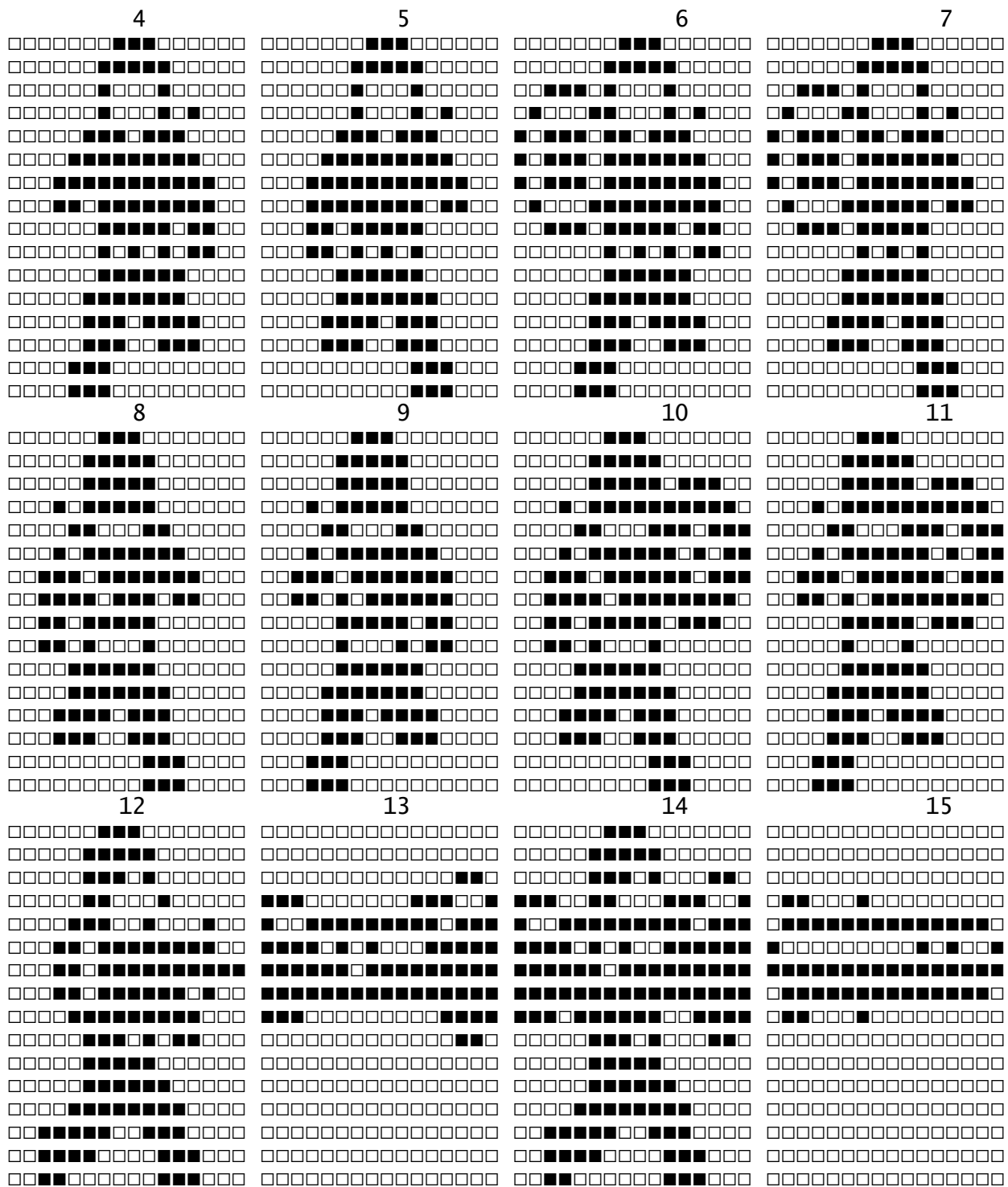


[illegible]



There are 16 sprites, 32 bytes each. This requires more than 6 lines of code, leaving almost no space for the game code:

```
0DATA0,0,0,0,255,255,0,0,0,0,0,0,0,0,0,0,0,0,239,239,0,0,0,0,0,0,0,0,0,0,
0,0,0,127,127,0,1,127,255,127,63,23,15,7,3,7,0,0,3,247,247,5,255,255,255,255,25
5,158,158,12,255,255,3,119,239,223,191,80,254,253,251,240,255,121,121,48,255,8,
200,255,221,221
1DATA255,85,255,255,255,127,255,231,231,195,255,0,0,224,192,176,48,0,255,255,25
5,255,255,158,159,14,252,0,0,0,0,0,6,15,239,246,248,240,192,128,0,0,0,1,3,2,2,7
,15,31,27,3,2,3,7,7,7,14,14,192,224,32,40,112,248,252,252,236,172,240,240,120,5
6,0,0,1,3,2,2,7
2DATA15,31,31,27,26,7,7,15,14,0,0,192,224,32,40,112,248,252,236,224,160,224,240
,112,112,56,56,1,3,58,70,187,187,187,71,59,2,3,7,7,7,14,14,192,224,32,40,112,24
8,252,252,236,172,240,240,120,56,0,0,1,3,58,70,187,187,187,71,59,2,7,7,15,14,0,
0,192,224,32,40
```

```

3DATA112,248,252,236,224,160,224,240,112,112,56,56,3,7,7,23,12,23,59,61,55,52,1
5,15,30,28,0,0,128,192,192,192,96,240,248,216,192,64,192,224,224,224,112,112,3,
7,7,23,12,23,59,53,7,4,7,15,14,14,28,28,128,192,192,192,96,240,248,248,216,88,2
24,224,240,112,0
4DATA0,3,7,7,23,12,23,59,61,55,52,15,15,30,28,0,0,128,192,220,254,119,235,247,2
54,220,64,192,224,224,224,112,112,3,7,7,23,12,23,59,53,7,4,7,15,14,14,28,28,128
,192,220,254,119,235,247,254,220,64,224,224,240,112,0,0,3,7,7,6,14,27,27,27,15,
7,7,7,15,62,60,48
5DATA128,192,64,32,68,252,255,244,248,88,192,224,240,112,56,56,0,0,0,224,159,24
5,253,255,224,0,0,0,0,0,0,0,0,6,57,247,31,255,255,15,6,0,0,0,0,0,3,7,7,230,
159,245,253,255,239,7,7,7,15,62,60,48,128,192,70,57,247,63,255,255,207,70,192,2
24,240,112,56,56
6DATA0,0,0,98,127,128,255,127,98,0,0,0,0,0,0,0,0,0,0,0,254,41,255,254,0,0,0,0,0
,0,0,0

```

The first optimization is getting rid of 0s, since MSX BASIC allows empty DATA instead:

```

0DATA,,,,,255,255,,,,,,,,,,,,,239,239,,,,,,,,,,,,,127,127,,1,127,255,127,63,
23,15,7,3,7,,,3,247,247,5,255,255,255,255,255,158,158,12,255,255,3,119,239,223,
191,80,254,253,251,240,255,121,121,48,255,8,200,255,221,221,255,85,255,255,255,
127,255,231,231
1DATA195,255,,,224,192,176,48,,255,255,255,255,255,158,159,14,252,,,,,6,15,239
,246,248,240,192,128,,,,1,3,2,2,7,15,31,27,3,2,3,7,7,7,14,14,192,224,32,40,112,
248,252,252,236,172,240,240,120,56,,,1,3,2,2,7,15,31,31,27,26,7,7,15,14,,,192,2
24,32,40,112,248
2DATA252,236,224,160,224,240,112,112,56,56,1,3,58,70,187,187,187,71,59,2,3,7,7,
7,14,14,192,224,32,40,112,248,252,252,236,172,240,240,120,56,,,1,3,58,70,187,18
7,187,71,59,2,7,7,15,14,,,192,224,32,40,112,248,252,236,224,160,224,240,112,112
,56,56,3,7,7,23
3DATA12,23,59,61,55,52,15,15,30,28,,,128,192,192,192,96,240,248,216,192,64,192,
224,224,224,112,112,3,7,7,23,12,23,59,53,7,4,7,15,14,14,28,28,128,192,192,192,9
6,240,248,248,216,88,224,224,240,112,,,3,7,7,23,12,23,59,61,55,52,15,15,30,28,,,
,128,192,220,254
4DATA119,235,247,254,220,64,192,224,224,224,112,112,3,7,7,23,12,23,59,53,7,4,7,
15,14,14,28,28,128,192,220,254,119,235,247,254,220,64,224,224,240,112,,,3,7,7,6
,14,27,27,27,15,7,7,7,15,62,60,48,128,192,64,32,68,252,255,244,248,88,192,224,2
40,112,56,56,,,
5DATA224,159,245,253,255,224,,,,,,6,57,247,31,255,255,15,6,,,,,3,7,7,230,
159,245,253,255,239,7,7,7,15,62,60,48,128,192,70,57,247,63,255,255,207,70,192,2
24,240,112,56,56,,,98,127,128,255,127,98,,,,,,,,,254,41,255,254,,,,,,,,,

```

Now six lines – still not a plenty of code space. We need a small compression algorithm. Consider the left top corner of the first sprite:

□□□□□□□□	0	□□□□□□□□	0
□□□□□□□□	0	□□□□□□□□	0
□□□□□□□□	0	□□□□□□□□	0
□□□□□□□□	0	□□□□□□□□	0
■■■■■■■■■	255	■■■■■■■■■	239
■■■■■■■■■	255	■■■■■■■■■	239
□□□□□□□□	0	□□□□□□□□	0
□□□□□□□□	0	□□□□□□□□	0
□□□□□□□□	0	□□□□□□□□	0
□□□□□□□□	0	□□□□□□□□	0
□□□□□□□□	0	□□□□□□□□	0
□□□□□□□□	0	□□□□□□□□	0
□□□□□□□□	0	□□□□□□□□	0
□□□□□□□□	0	□□□□□□□□	0
□□□□□□□□	0	□□□□□□□□	0
□□□□□□□□	0	□□□□□□□□	0

It seems perfect for RLE encoding: 4x0, 2x255, 14x0, 2x239, 10x0. We will encode data as:

256 * (count - 1) + value

giving 768,511,3328,495,2304.

It is clear that many sprites differ only in details, notably the Ukrainian soldier:

Previous		Current
□□□□□□■	1	□□□□□□■
□□□□□□■	3	□□□□□□■
□□□□□□■	2	□□□□□□■
□□□□□□■	2	□□□□□□■
□□□□□■	7	□□□□□■
□□□■	15	□□□■
□□■	31	□□■
sprite #4		sprite #5

If some bytes match to the corresponding ones in the previous sprite, we encode data as:

-(count - 1)

giving -6.

Using the above technique, our sprite sheet transforms into:

```
0DATA768,511,3328,495,2304,-3,383,,1,127,255,127,63,23,15,7,3,7,-1,3,503,5,1279
,414,12,255,255,3,119,239,223,191,80,254,253,251,240,255,377,48,255,8,200,255,4
77,255,85,-2,127,255,487,195,255,256,224,192,176,48,,1279,158,159,14,252,1024,6
,15,239,246,248
1DATA240,192,128,512,1,3,258,7,15,31,27,3,2,3,519,270,192,224,32,40,112,248,508
,236,172,496,120,56,-1,-6,31,27,26,263,15,14,256,-6,236,224,160,224,240,368,312
,-1,58,70,699,71,59,2,3,519,270,-6,252,236,172,496,120,56,256,-9,263,15,14,256,
-6,236,224,160
2DATA224,240,368,312,3,263,23,12,23,59,61,55,52,271,30,28,-1,128,704,96,240,248
,216,192,64,192,736,368,-6,53,7,4,7,15,270,284,-6,248,216,88,480,240,112,256,-6
,61,55,52,271,30,28,256,-1,220,254,119,235,247,254,220,64,192,736,368,-6,53,7,4
,7,15,270,284,-9
3DATA480,240,112,256,-2,6,14,539,15,519,15,62,60,48,-1,64,32,68,252,255,244,248
,88,192,-2,312,512,224,159,245,253,255,224,2048,6,57,247,31,511,15,6,1280,3,263
,230,-3,239,519,15,62,60,48,128,192,70,-1,63,-1,207,70,192,224,240,112,312,512,
98,127,128,255
4DATA127,98,2560,254,41,255,254,1792
```

That is slightly more than 4 lines. The purified decoder is:

```
0DEFINT A-Y:FOR I=0 TO 15:S$="":FOR J=1 TO 32:READ N:IF N<0 THEN N=-N:S$=S$+MID$(SPRITE$(I
-1),J,N+1)ELSE V=N AND 255:N=N/256:S$=S$+STRING$(N+1,V)
1J=J+N:NEXT J:SPRITE$(I)=S$:NEXT
```

162 characters in total, so we saved about two lines for the meaningful code!

LISTING

```
0DEFINT A-Y:D=0:T=0:SOUND 7,0:COLOR 3,1,1:SCREEN 5,2:CLS:READ T$,U$,V$:OPEN"GRP:"AS1
:PSET(112,92),0:?"#1,T$:FOR I=0 TO 15:S$="":FOR J=1 TO 32:READ N:PSET(ABS(N),12*I),12:I
FN<0 THEN N=-N:S$=S$+MID$(SPRITE$(I-1),J,N+1)ELSE V=N AND 255:N=N/256:S$=S$+STRING$(
N+1,V):DATANLAW:
1J=J+N:NEXT J:SPRITE$(I)=S$:Z=I*ATN(1)/8:M=20*SIN(Z):FOR J=16*COS(Z) TO 191 STEP 5+N:
PSET(M,J),12:NEXT:NEXT I:PSET(112,92),0:COLOR 1:?"#1,T$:PUTSPRITE 8,(0,0),3,13:PUTS
PRITE 9,(0,176),3,13:FOR I=1 TO 5:READ M,N,F,C:LINE(M,N)-(M+11,N+F),C,BF:NEXT:C=3:N=
0:P=88:X=255:Y=P:
2J=STICK(0):F=(P AND 4)=0:PUTSPRITE 4,(0,P),C,12+(8+F)*(J=5)-N+(4+F)*(J=1)-N:P=P+4
*(J=1 AND P>0)-4*(J=5 AND P<176):N=(N OR P=0 OR P=176) AND D<16:PUTSPRITE 5,(D,M),14 AND D>0
,15:SOUND 8,D>0 AND 7:D=D>0 AND (D+8) AND 255:M=M+(D>=16 AND D<=112):IF STRIG(0) AND N AND D=
0 THEN N=0:M=P:D=8:
3I=X:FOR J=0 TO 3:K=I<=255 AND I>=-32:PUTSPRITE J,(I,Y),-K-K,J:I=I+16:NEXT:PSET(X+32,
Y+15+2*RND(1)),-12*(RND(1)>.5):X=X-T/10-1:F=X>-16 OR ABS(Y-P)>8:C=C AND F:IF X<-64 TH
```

```

ENPLAYU$:FORJ=0TO1:FORI=193TO198:LINE(2,I)-(13,I),POINT(242,I):NEXT:J=-STRIG(0)
:NEXT:RUN:'Glory:
4IFD<X+29ORD>X+51ORM<Y-12ORM>YTHEN2ELSEPUTSPRITE5,,0:FORR=0TO15:CIRCLE(D+8,M+6)
,R,8:SOUND8,R:NEXT:FORR=15TO0STEP-1:J=RAND3:PUTSPRITEJ,(16*J+X,Y),R/4+8,J:CIRCL
E(D+8,M+6),R,1:NEXT:D=0:X=255:Y=160*RND(1)+16:T=T+1:LINE(35,193)-(6*T+35,198),8
,BF:IFT<31THEN 2:
5PLAYV$:FORJ=0TO3:PUTSPRITEJ,,0,J:NEXT:FORJ=0TO1:FORI=193TO198:LINE(242,I)-(253
,I),POINT(2,I):NEXT:J=-STRIG(0):NEXT:RUN:DATAV15L8GEL4AL2E,V15L8CL4FEGL8CFGABL4
C5L8F,768,511,3328,495,2304,-3,383,,1,127,255,127,63,23,15,7,3,7,-1,3,503,5,127
9,414,12,255:'UA:
6:DATA255,3,119,239,223,191,80,254,253,251,240,255,377,48,255,8,200,255,477,255
,85,-2,127,255,487,195,255,256,224,192,176,48,,1279,158,159,14,252,1024,6,15,23
9,246,248,240,192,128,512,1,3,258,7,15,31,27,3,2,3,519,270,192,224,32,40,112,24
8,508,236,172,496
7DATA120,56,-1,-6,31,27,26,263,15,14,256,-6,236,224,160,224,240,368,312,-1,58,7
0,699,71,59,2,3,519,270,-6,252,236,172,496,120,56,256,-9,263,15,14,256,-6,236,2
24,160,224,240,368,312,3,263,23,12,23,59,61,55,52,271,30,28,-1,128,704,96,240,2
48,216,192,64,192
8:DATA736,368,-6,53,7,4,7,15,270,284,-6,248,216,88,480,240,112,256,-6,61,55,52,
271,30,28,256,-1,220,254,119,235,247,254,220,64,192,736,368,-6,53,7,4,7,15,270,
284,-9,480,240,112,256,-2,6,14,539,15,519,15,62,60,48,-1,64,32,68,252,255,244,2
48,88,192,-2,312:
9:DATA512,224,159,245,253,255,224,2048,6,57,247,31,511,15,6,1280,3,263,230,-3,2
39,519,15,62,60,48,128,192,70,-1,63,-1,207,70,192,224,240,112,312,512,98,127,12
8,255,127,98,2560,254,41,255,254,1792,2,193,2,5,2,196,2,11,242,193,1,15,242,195
,1,4,242,197,1,8:

```

EXPLANATION

Line 0: Screen and variable setup. Sprite decoder (described above) intermixed with the field drawing code, see PSET. D is the horizontal missile coordinate, T is the number of tanks destroyed.

Line 1: Sprite decoder (described above) intermixed with the trench drawing code, see PSET. Draw both NLAW supplies. Draw Ukrainian and russian flags. Initialize variables. C is the player color/visibility, N is the NLAW flag, P is the player's vertical coordinate, X and Y are tank coordinates.

Line 2: Poll joystick (i.e., arrow keys). F is the player step. Draw the player sprite according to direction, step, and NLAW presence. Modify the player position. Draw the missile sprite (transparent if not launched). Play the sound effect. Modify NLAW coordinates D and M. Poll the button (i.e., space bar). Launch the missile if allowed.

Line 3: Draw the T-90 tank (4 sprites) and its trail (see PSET). Modify the tank's position. F is the tank/player collision, hide the player if so. If the left side is approached then play the Game Over music, replace the Ukrainian flag, and wait until [SPACE] is pressed, then restart.

Line 4: Loop to 2 if tank is not hit by the missile. Otherwise hide the missile, draw the explosion (see CIRCLE), and play the sound effect. Burn the tank (i.e., change colors in a loop). Initialize next tank's coordinates. Update the progress bar, loop to 2 if there are tanks left.

Line 5: Play the Victory march music, raise the Ukrainian flag, and wait until [SPACE] is pressed, then restart. DATA for Game Over and Victory music. Sprites (begin).

Line 6: Sprites (continue).

Line 7: Sprites (continue).

Line 8: Sprites (continue).

Line 9: Sprites (end). Ukrainian and russian flag stripes (starting 2,193,2,5) - x, y, height - 1, color.