Bitmap Graphics - Size (Compression) versus Speed (Decompression) - Demo picture Barbarian by helpcomputer0

zx7 dan3 lzf pletter rle compression colecovsion speed size compare dan1 bitbuster

By newcoleco
August 7 in ColecoVision Programming

Reply to this topic

newcoleco
Posted August 7

In order to explain myself better about graphics data compression and the various options I've tried. Here are 5 formats, all decompression possible in VRAM, all in a single project, source code included. Enjoy.

The sample source picture is from ZX Spectrum Pixel Art made by someone named Adam I believe, username helpcomputer0.

I converted the picture to ColecoVision graphics using my SCR2PC tool made a long time ago.

The 5 data compressions tested here are : DAN3, ZX7, PLETTER, LZF for ZX-Spectrum, RLE

I simply used my recent all-in-one data compression tool set and gave it a go.

I quickly put a slideshow of the same picture compressed in these different formats and gave it a go to see their decompression speed.

In the slide show I believe I've put the images in order of size, ZX7 and Pletter for this case gave the exact same data size, the difference is in the decompression routine size. DAN3 is the most complex decompression routine and yet still decompress fine.

Two things to notice. If you want better than RLE size and yet still think the LZSS variants including DAN3, ZX7 and Pletter are too slow, LZF is a valid solution.

Enjoy!

The source code barbarian_src.zip
The ROM file barbarian.rom

Edited August 8 by newcoleco

Quote

2

newcoleco
Posted yesterday at 08:29 PM

Added ZX0 to VRAM to the mix since there was space for it in this 32KB typical ColecoVision ROM file.

This tiny project contains :
Pletter to VRAM
ZX7 to VRAM : This format was develop by Einar Saukas in 2012 for the ZX Spectrum homebrew scene, ported to ColecoVision scene by newcoleco in 2013.
ZX0 to VRAM : This format was develop by Einar Saukas & Urusergi in 2021 as a replacement for einar's zx7, ported to ColecoVision scene by newcoleco in 2025.
DAN3 to VRAM : This format was develop be newcoleco in 2018 as a last effort to improve the data compression toolkit inside the newcoleco's coleco devkit.
LZF to VRAM : This version of LZF had an end-marker which I believe is the same format used in the ZX Spectrum homebrew scene, ported to ColecoVision scene by newcoleco in 2025.
RLE to VRAM : This RLE format was established by Marcel deKogel in 1996 and used in his coleco devkit which was used by newcoleco for my own popular cv devkit to program new ColecoVision games in C language early 2000.

ZX0 <- size
code size : 135 bytes
pattern : 3545 bytes
color : 684 bytes
total : 4364 bytes
time : 0.96 seconds

ZX7 <- size
code size : 125 bytes
pattern : 3604 bytes
color : 735 bytes
total : 4464 bytes

time : 1.01 seconds

DAN3 <- 🥉 size
code size : 194 bytes
pattern : 3579 bytes
color : 707 bytes
total : 4480 bytes
time : 1.07 seconds

PLETTER <- 🥉 speed
code size : 201 bytes
pattern : 3606 bytes
color : 733 bytes
total : 4540 bytes
time : 0.92 seconds

LZF <- 🥈 speed
code size : 106 bytes
pattern : 3982 bytes
color : 822 bytes
total : 4910 bytes
time : 0.86 seconds

RLE (MDKRLE) <- 🥇 speed
code size : 37 bytes
pattern : 4658 bytes
color : 845 bytes
total : 5540 bytes
time : 0.41 seconds

That is the results for the barbarian bitmap screen.
Barbarian Slideshow ROM file : 📎 barbarian.rom
Barbarian Slideshow SRC file : 📎 barbarian_src.zip

Note : A different image or more than one image is enough to change the results, especially regarding size.

Edited yesterday at 09:32 PM by newcoleco

alekmaul
Posted yesterday at 11:07 PM

thanks a lot Amy, I will check the source code for ZX0 and ZX7 which seem nice regarding size of compression 😮 !

Edited yesterday at 11:07 PM by alekmaul

Quote

1

newcoleco
Posted 13 hours ago

> **On 8/29/2025 at 11:07 PM, alekmaul said:**
>
> thanks a lot Amy, I will check the source code for ZX0 and ZX7 which seem nice regarding size of compression 😮 !

ZX0 is using Markov Chain and interlaced Elias Gamma encoding.
The idea is quite simple.
In LZSS, a single bit is needed to tell if the token represents literals or a match.
But ZX0 wanted a special case where you can avoid encoding an already seen offset for a match.
And having two match in a row with the same offset is making no sense unless forced by size match limitation.
But if a match follow a bunch of literals it may be possible that the match can use the same offset value of the previously seen match.
Markov Chain tells then that by default the previous seen match was an offset of one and the compression stars with encoding literals.
After a token of literal, two choices only, a match with already seen

offset or a match with a new offset to encode.
After a match, two choices, a match with a new offset to encode or literals.
Markov Chain is simply saying that you need only 1 bit based on the context, so why not following the context logically rather than encoding it like a bit saying it is a match and a bit for an already seen offset.

I can see one way to potentially improve on ZX0 compression result that might also make some cases decompress faster by just changing Elias Gamma with the encoding I use in DAN3 that is more a exp-Golomb kind of encoding reducing the number of bits needed for higher values every time. But that would need testing to see if I am right.

And you can also try all these data compression at once here online if your computer is good : RetroCompress Lite - Classic 8-bit Compression Algorithms

Try compressing data while keeping in mind your goal (size versus speed) then add the size of the decompression routine and you should know the size of the ROM used by your graphics with the associated decompression method.

Working on adding DAN1, available since 2016 in my toolkit, but this Barbarian picture do not compress enough to fit all these data decompressions to VRAM in a single ROM.
I might have to do another try with another image, not coming from ZX Spectrum pixel art gallery this time.

Edited 13 hours ago by newcoleco

newcoleco
Posted 44 minutes ago

What's new?

+ Picture called "Warrior" by Adam (helpcomputer0), same author as the "Barbarian" picture.

+ DAN1 (size 205 bytes) and BitBuster v1.2 (size 131) decompression.

Data compression results (just for the graphics)
ZX0 : 2458 + 391 = 2849
DAN3 : 2500 + 391 = 2891
DAN1 : 2504 + 399 = 2903
ZX7 : 2569 + 415 = 2984
Pletter : 2572 + 416 = 2988
BitBuster : 2579 + 423 = 3002
LZF : 2749 + 446 = 3195
RLE : 2981 + 706 = 3687

Data + Decomp results
ZX0 : 2849 + 135 = 2984
DAN3 : 2891 + 194 = 3085
DAN1 : 2903 + 205 = 3108
ZX7 : 2984 + 125 = 3109
BitBuster : 3002 + 131 = 3133
Pletter : 2988 + 201 = 3189
LZF : 3195 + 106 = 3301
RLE : 3687 + 37 = 3724

As you can see, with a simpler image with lots of empty space, the order changed.
And the big surprise for me is BitBuster winning against Pletter even with the extra 4 bytes header per file and 4 bytes in the decomp routine to skip that header just because of the decompression method size and only a single tiny compressed image.

The source code : warrior_src.zip
The ROM file : warrior.rom

Quote

Join the conversation

You can post now and register later. If you have an account, sign in now to post with your account.
Note: Your post will require moderator approval before it will be visible.

Reply to this topic...

Share Followers 0

Go to topic listing

Recently Browsing 1 member

Captain Cozmos

AtariAge

Home
2600
5200
7800
Lynx
Jaguar

Forums

Forum Home
Clubs
Blogs
Gallery
Subscriptions

Store

Store Home
New Products
Bestsellers
Gift Certificates

General

Contact
Submit News
Privacy Policy
Legal

Follow AtariAge

Like us on Facebook
AtariAge Facebook Group
Follow us on Twitter
AtariAge on Instagram
Watch us on YouTube
Chat on Discord