

Game Loader COMMODORE 64 ATARI 2600 Adaptor

White Paper From RelationalFramework.com

Introduction

Game Loader is an Atari 2600 Emulator for the Commodore 64 that adds fantastic Commodore Graphics to Atari 2600 games!

Background

C64 – The Commodore 64 is the most popular home computer of the 80's.

Atari 2600 – The Atari 2600 is the most popular video game console of the 80's.

Inspiration

Inspired by Jack Tramiel and the Vaporware add for Game Loader from 1983

Innovation Benchmarking

Game loader uses Innovative Technologies to achieve 1 MHz system emulation on 1 MHz target platform at full speed with cycle precise transition for side-by-side comparisons:

[Commodore 64 Atari 2600 Emulator Silly Venture Benchmarking \(relationalframework.com\)](http://relationalframework.com)



Technical Details

Game Loader recompiles the source and combines it with the emulator to create stand alone program binaries that can load from Tape, Disc or Cartridge on the Commodore 64.

Game Loader is currently compatible with CBS RAM and SuperCharger Atari games.

The high-performance soft ANTIC kernel has been ported at 100% speed and includes one demo of the kernel outperforming a real Atari 2600.

Hybrid emulation – Atari 2600 Shadow bus for C64 register cascades for real time emulation of the game loop combined with Kernel cores for each game or set of games that use the same kernel are added to the kernel library allowing cycle exact transition of even the most high-performance Atari 2600 games!

Greater compatibility - Playing Atari games that even the real hardware cannot:

Atari games that can only under CBS RAM or only under the SuperCharger memory formats are fully supported by the Game Loader Atari Adaptor. One example is the new SuperCharger Space Invaders that runs in Game Loader and under CBS RAM, but not on a real SuperCharger because it runs out of Memory.

Extra Game Loader Features

Adding custom PETSCII graphics to Atari games

Custom PETSCII graphics can be applied to the foreground and background tiles, comprised of 4 PETSCII characters each. These can be defined in the emulator as filter settings to be static bound, or can be dynamically changed at runtime by C64 side only feature set and commands per the PETSCII demo.

TIA sound emulation with SID

There are currently two TIA sound emulation routines implemented in the emulator, real time register manipulation wrapping the Shadow bus TIA registers to the SID which is supported in VICE and on the real C64, and more dynamic soft synth conversion in the Framework API which fails in VICE as “SID overflow”.

Status Working great in beta more to be done....

Emulating TIA sounds from the twin oscillators on the SID's is a 29 register cascade in real time.

Compatibility with Classic Atari Games – per Kernel CORE added.

Adding custom VIC-II cores – Custom cores for batari BASIC and batari BASIC SuperChip Atari games planned.

Future expansion ideas


Adding ARM and DPC Compatibility for enhanced Atari games like PITFALL II via an ARM expansion board like C64 Chameleon cartridge.

Game Loader on Other Systems – Atari 400/800/5200 was previously under development, NES may be next:


NOAC potential in Atari Flashback to run real Atari Games

The Atari Flashback initially used a Nintendo on a Chip and featured imitation Atari games that were really Nintendo games. With the Game Loader *technology for 1 MHz on 1 MHz emulation* this product could have succeeded. Existing Flashback 1 units can potentially be reloaded with the Game Loader emulator playing real Atari games on the NOAC!

Television Threading Model Implementation [coding to the CRT]

	<p>Architecture: The Atari 2600 <i>paces the beam</i> to program classic Television. Atari game loops run inside the vertical blanks.</p>  <p>Read <i>Racing the Beam</i> for more details on Game Loader Architecture</p>
--	---

The Soft ANTIC Blitter Kernel:

	<p>The Soft ANTIC Blitter Kernel functions like the Nintendo PPU to scroll a playfield Camera around a large tile mapped virtual world. It also allows the screen to be split into independent scroll zones each with it's own Camera just like Atari Home Computers as shown in the Display List Demo:</p> 
--	---

ATARI 2600 Assembly and BASIC games both supported:

	<p>Assembly programs and SuperCharger BASIC and Atari Flashback BASIC are supported allowing Cross development on the Atari 2600 and Commodore 64 platforms with an extended feature set available on the Commodore 64 for creating Super Atari 2600 games.</p> <p>The batari BASIC standard and 8K SC kernel are being ported next since these kernels will enable the largest volume of Homebrew games to run with no changes to the code while adding Fantastic Commodore graphics to the games!</p>
<p>Game Loader is compatible with Assembly and BASIC Atari 2600 Game Program Cartridges and Cassettes using CBS RAM and SuperCharger enhanced formats.</p> <p>.....</p> <p>Gameloader ouptputs standalone C64 prg binaries with the emulator and source combined allowing fantastic Commodore graphics to be put on the ROMS!</p>	

Simple classic BASIC Programming Example from the 2021 10 Liner contest <http://Basic10liner.com>

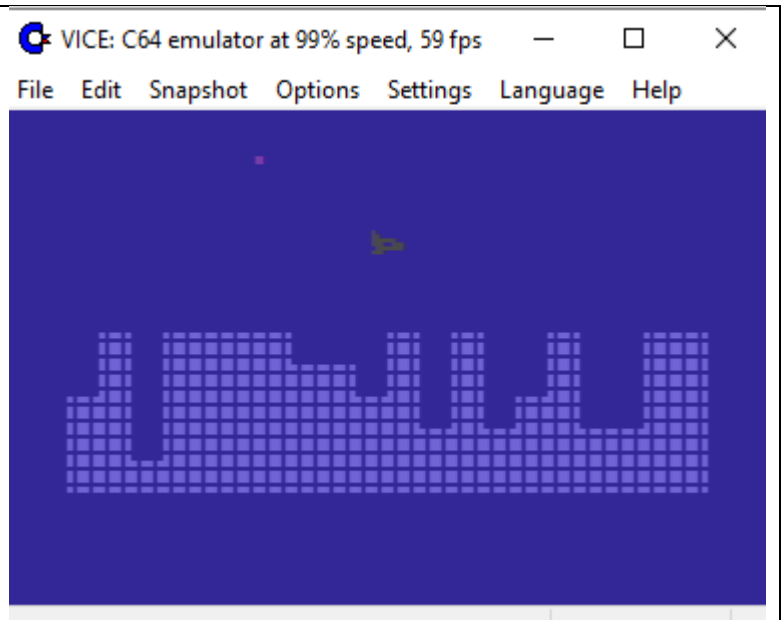
```

1 rem -----
2 rem --- SUPERBLITZ Throttle Control 3/2021
3 rem --- basic10liner.com competition verison
4 rem --- Waves and Missions
5 rem -----
6 rem --- Insignia on Plane revealed after Night Mission is completed!
7 rem --- Multiple waves over differently colored Sky's and Cities
8 rem --- More optical illusions: Some color combinations create unreal artifacting such as textured bricks -
9 rem --- complete several rounds and describe what you encounter!
10 rem --- note: Classic hardware and a CRT required to experience artifacting textures and additional illusions
11 rem --- unique features - you control your speed and difficulty increases as you play
12 rem --- BW switch pegs the throttle (Advanced)
13 rem --- Each wave achieved keeps it's colors in Attract mode!
14 rem --- Unlimited continues until powering off the console!
15
16 0 data city 5,9,6,9,7,6,7,5,9,5,5,5,6,6,7,5,8,5,8,7,5,8,8,5,5,6,6,7,5,7,8,9,8,8,7,8,9,5,6,8,5,9,6,6,7,5,7,5,5,8,5
17 1 if g=0 or CXP0FB>126 then CCLR=0:for j=0 to 9:player1(j)=189:player0(j)=pl(j):rowcolors(j)=178+w:next j else goto 3
18 2 for j=20 to 71:k=j-20:k=city(k):for i=k to 9:vwpxel(j,1,0):next i,j:player0y=96:player0x=84:y=11:g=1:w=16*z
19 3 if f<player0y/52 and joy0right=0 and SWCHB[247=255 then f=f+1:goto 9 else AUDC1=7:AUDF1=BITIndex/5:AUDV1=BITIndex:f=0
20 4 AUDV0=F:scrollvirtualworldtoggle=1:BITIndex=BITIndex+1:missile0x=missile0x+2:data p1 0,224,127,231,252,192,128,0
21 5 if joy0fire=1 and y>8 then AUDF0=12:AUDC0=9:SUSTAINFORFRAMES=7:x=BITIndex+9:i=96-player0y:i=i/10:y=i+1:remPlayahTune
22 6 if y<10 then vwpxel(x,y,bindplayer1):COLUP1=M(y):y=y+1:data M 122,138,12,170,154,250,234,218,202,186,42,58,74,28
23 7 if y<10 and vwpxel(x,y,p01)>0 then vwpxel(x,y,flip):player1x=0:player1y=0:AUDC0=y:y=21:AUDF0=4:AUDV0=15:rem Hit!
24 8 if BITIndex>71 then BITIndex=0:player0y=player0y-2 else missile1x=missile1x+1:missile1y=missile1y+3:rem SUPERBLITZ 86
25 9 if player0y=0 then g=0:player0colors(3)=14:player0colors(4)=112+z:COLUBK=M(z)-10:z=z+1 else missile0y=missile0y+2
26

```

Classic vintage BASIC Programming with line numbers and no extra graphical workspace structures like in the 80's is supported and can access the NES PPU style functions of the soft ANTIC Blitter kernel. You can see from the simple BASIC listing above the Atari 2600 registers are dynamically manipulated in the source code.

None of this code is changed because the emulator sub framework bound to the prg binary includes an Atari 2600 Shadow bus full of these Shadow registers as part of the innovation for real time emulation of 1 MHz systems on a 1 MHz platform with B-tree style balancing of the TTM Television Threading Model.



Applying Fantastic Commodore PETSCII Graphics:

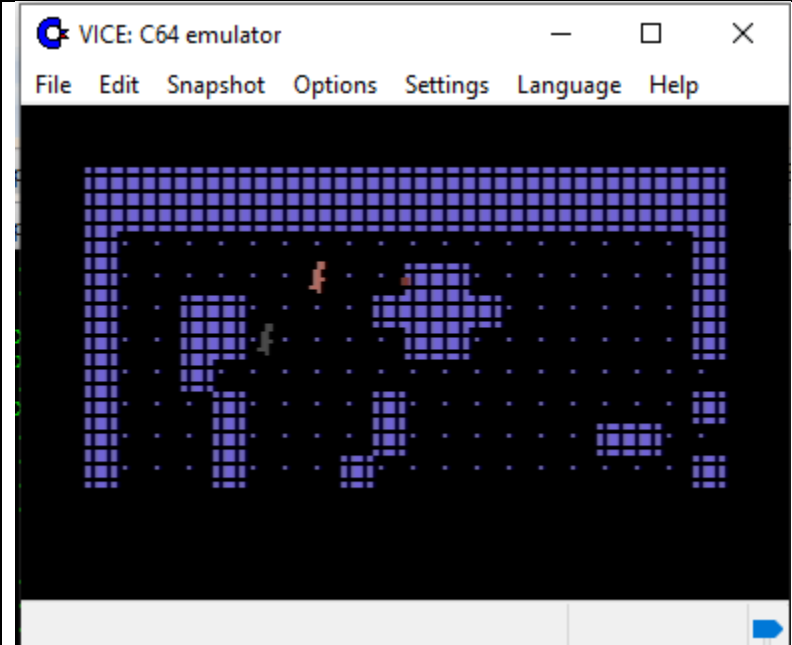


The two commented lines below the PETSCII Symbols keyboard illustrate how easy it is to dynamically redefine the four PETSCII graphics characters in the Foreground and Background tiles while the program is running. Note the lines are commented because they will not work on a real Atari 2600 but can be used to enable Super Atari games for the Commodore 64 with extra functionality! The PETSCII video on the benchmarking page shows a programmatic

```
rem c64 specific tile changeup:
rem for x=0 to 3:BackgroundTileCharacters(x)=125:next
```

example of an Atari 2600 "Super program".

Applying PETSCII filters in Game Loader may create new versions of existing Atari games:



The screenshot shows the VICE C64 emulator window. The title bar reads "VICE: C64 emulator". The menu bar includes "File", "Edit", "Snapshot", "Options", "Settings", "Language", and "Help". The main display area shows a game with a blue grid overlay, likely representing the PETSCII filter. The game appears to be a maze or platformer with a character and various obstacles.

Adding Stars and Stripes to BERSERK2000 R3 changes the game because the sup playfield pixel elements interact *like tiny pieces of playfield "semigraphic shrapnel"* adding new functionality to Super Atari games never before seen!

Sometimes a variation or an entirely new genre of Atari 2600 Super game may emerge just from the static application of Fantastic Commodore graphics in Game Loader emulators PETSCII settings screen for the C64 prg binary!

Complex BASIC and Assembly Example

BERSERK2000 R3 became Berserk Stars and Strips with no changes to the code! How'd that happen? Here's the complex BASIC program listing where soft ANTIC display lists are used as well as graphical ASCII art to "draw" the virtual world and the sprite definitions, and a Tracker/Sequencer chip tune musical score is also present at the end of the listing.

The format is very familiar for batari BASIC programmers except for the simple Tracker/Sequencer which shapes the notes for interesting sounds on the Atari or C64 and is very simple and intuitive to use! The playfield CAM and Display Lists are of course also new to bB programmers and allow speeds rivaling Assembly.

Inline Assembly and BASIC enhancement features

Inline Assembly is slightly different than in batari BASIC and the ":" concatenator operator enables horizontal (wider) Assembly code design. You will see two gameloops compared to one in batari BASIC to allow more processing time and dynamic balancing. There is also a "Kitchen Sink" section for more time if the Video Signal is changed (the soft ANTIC Blitter Kernel can output 30 Hz or several variable Hz rates, but this game is at the normal non-standard 60 Hz Atari signal):

Program listing with no changes to the code where just the PETSCII filter creates a new game:

```
rem -----
rem SuperCharger BASIC Program BERZERK 2000, a BERZERK scroller reimaged!
rem -----
rem Compatible with the Atari Flashback Portable and all Atari consoles!
rem (Cross compiles under Atari Flashback BASIC or SuperCharger vwBASIC)
rem -----
rem Brought to you by RelationalFramework.com (the best IDE for SQL Server!)
rem -----
rem version history:
rem v1    smooth running man in a scrolling maze world
```

```

rem v1.1 computer opponents added (no AI) with N smart multiplexing character
sprites
rem Rolled back to v1 to used motion blur reduction and 4 character sprites with
even multiplexing
rem 20190808 MBR CAM mechanics - board scrolling 2 tile pixel bursts (vertical)
and (horizontal)
rem 20190809 Adding AI for 3 balanced character opponents (replaces variable
flicker)... done
rem color map board for scrolling - Done >> color sections scroll,
colors selected from https://www.randomterrain.com/atari-2600-memories-tia-color-
charts.html
rem 20190809 Add AI notes ... !AI players can change dir (delay) each frame turn
if a wall is encountered! (saves cycles and adds escape delay for the human
player)
rem 20190812 AI players 2 and 3 are reusable - excellent! Only 400 bytes free;
repeat this opt:
rem make AI player 1 similarly reuse variables and routines with human
player... WIP
rem Smart shots should follow players like Robots do WIP
rem making direction/bounce routine reusable
rem 20190814 removed distortion; tracker changes the envelope every frame
rem adding ricochet shots
rem 20190821 with Chiptune III - catchy synth fx tune
rem 20190828 Also adding more intelligent maze rebuilding based on blue & green
robots row for the Y tile axis (z)...
rem 20190828 Added subtune when clone regenerates
rem 20190829 Fixed players ghosting semi-static on top row
rem 20190829 Fixed players ghosting to the left when leaving the camera view on
the right
rem 20190830 Added red and green pulse waves (not flashes) from enemy or friendly
fire impact
rem -----
rem -----
rem 20211207 Automated tandem control with the CPU for computer assisted play or
demo
rem 20211207 including as bonus hidden demo in SillyVenture2021 entry (press
button when "Party in Gdansk!" appears to play! :)
rem 20211207 Added PlusCart/UnoCart CTRLPF=0 fix to turn off reflective
playfields for loads other than main
rem -----
rem -----

rem Controls:
rem -----
rem --- Joystick - Player
rem --- select/reset - change color schemes
rem --- bw - Otto (difficult)
rem --- left difficulty - Advanced/Beginner

rem --- vars
rem --- i,j,k,l loop vars
rem --- t toggle, x,y (player0 x,y virtual world coordinates)
rem --- m,n,o incremental smooth movement counters
rem --- s kickout for fast movement during 30 FPS scrolling
rem --- r,q 30 FPS scroll burst x/y duration
rem --- px,py human player0 x,y fine coordinates
rem --- x,y,e human player0 sprite x,y tile coordinates and direction
rem --- u,v player0 sprite mirror fine coordinates
rem --- bx,by,g player0 sprite mirror x,y tile coordinawtes (DIR 1st AI player)
rem -- freed - var2,score player1 sprite x,y fine coordinates
rem -- w,z,p player1 sprite x,y tile coordinates and direction (next robot
character is reloaded from RAM array)
rem --- ??Laserbeam tile coordinates, ??>0 is the trip
rem --- ?? Laserbeam direction
rem --- h AI pointer

```

```

rem -- var1,var2,score,f > missile0/missile1 direction and duration variables
(next ball character is reloaded from RAM array)

rem ---init section, runs once: -----
a=0:gosub loadvirtualworld:rem playable Flashback version blue and green robots
are frozen, why?
rem SUPERCHARGERID=99:rem bonus playable demo for SillyVenture2021 competition
on 20211212!
CTRLPF=0:rem PlusCart/UnoCart fix, turns off reflective playfields for loaded
programs other than main
rem BITIndex=0
rem loadplayer0(0)
rem bind sprite to semigraphics virtualworld tile:
x=3:y=3: rem ***** player 0 virtualworld tile coordinates
bx=5:by=5:rem g=5: rem player0 sprite mirror x,y,direction (1st AI player init
and dir)
rem w=3:z=3:p=2: rem **** player 1 sprite x,y,direction (2nd AI player init and
dir)
w=vars(0):z=vars(1):o=vars(2):p=vars(3)
rem -----
rem ---gameloop subroutine, runs every frame: ---
rem BYTERowoffset=120
rem -----

rem -- piggyback missile1 propegation for relfecting shots
rem --- propegate players bouncing shot if it is in progress:
if f=0 then missile1y=0:goto skipshot1a
rem check for collision with the wall and bounce
if CXM1FB >126 then AUDC0=12:AUDV0=7:gosub nextdir:score=robotAI(h)
if score=1 then missile1y=missile1y+1:missile1x=missile1x-1
if score=2 then missile1y=missile1y+1
if score=3 then missile1y=missile1y+1:missile1x=missile1x+1
if score=4 then missile1x=missile1x-1
if score=5 then missile1x=missile1x+1
if score=6 then missile1x=missile1x-1:missile1y=missile1y-1
if score=7 then missile1y=missile1y-1
if score=8 then missile1x=missile1x+1:missile1y=missile1y-1
skipshot1
f=f-1

skipshot1a
rem clear collision register

rem -- e direction : 1 2 3
rem                  4 0 5
rem                  6 7 8

rem -- end piggyback missile0

rem --- 2x speed check console switch:if SWCHB|%11110111<>255 then
scrollvirtualworldtoggle=35:gosub DLI:scrollvirtualworldtoggle=0:goto 0:rem
double speed - every frame
rem Rollback, no DLI's this game: if t=0 then scrollvirtualworldtoggle=33:gosub
DLI:scrollvirtualworldtoggle=0:return: rem run this DLI every other frame

rem -- cycle player0 and player1 sprites at 30 hz each
if t=0 then goto DoPlayer0

rem -----
-----
rem -- AI Opponent #2 (player 1 Mirror #1) -----
-----
rem -----

```

```

rem pull for first robot call:
w=vars(0):z=vars(1):o=vars(2):p=vars(3):player1x=vars(4):player1y=vars(5)
rem 3,3,0,2,0,0
COLUP1=$BA:rem solid color sprite

AIOpponentTwo rem -- reusable routine can be overloaded for #3 called from
bottom blank if vars are swapped!

rem check for collision with AI Robot 2 or 3 and send it back home and deflect
the ball but don't destroy it:
rem 20190830 if CXMOP >126 then AUDCO=12:AUDVO=12:w=0:z=3:p=5:o=0:gosub
nextdir:var2=robotAI(h)
if CXMOP >126 then AUDVO=player0x:w=0:z=3:p=5:o=0:gosub nextdir:var2=robotAI(h)

rem -- 20190810 tile mapped 4 30 Hz balanced opponents, adding AI ...
rem -- relocating player0 mirror sprite AI to bottom vertical bank for load
balancing (when t=1)

rem --
rem mirror for sprite 1: todo after adding...
rem ??=player1x:??=player1y: rem hold player 1s xy (on overload do this!)
rem 20190810
rem shadowcopy call at end handles storing, full restore is here: player1x=var2:
player1y=score: rem restore player 1 mirror's x,y

rem --- cycle player animation
rem too slow i=o/2:i=i+b
" lda n: lsr: sta i: bcc fastercalcdone2: inc i:fastercalcdone2 "
i=spritemap(i):loadplayer1upsidedown(i)
rem loadplayer1upsidedown(o)
rem establish direction of tile based move if m is free
if p=1 or p=4 or p=6 then REFP1=255 else REFP1=0
if o>0 then goto skipdirection2ax

vwpixel(w,z,bindplayer1): rem binds the completed move after the micro
increments (or the initial move)
COLUBK=0

rem -----
rem -- e [p!] > direction:  1 2 3  -----
rem                          4 0 5  -----
rem                          6 7 8  -----
rem -----
rem                          6 7 8

i =w:j=z: rem preserve x and y for polling potential target square rollback
if p=1 then o=8:z=z-1:w=w-1:goto skipdirectionax
if p=3 then o=8:z=z-1:w=w+1:goto skipdirectionax
if p=2 then o=8:z=z-1:goto skipdirectionax
if p=6 then w=w-1:z=z+1:o=8:goto skipdirectionax
if p=8 then z=z+1:w=w+1:o=8:goto skipdirectionax
if p=7 then z=z+1:o=8:goto skipdirectionax
if p=4 then o=8:w=w-1:goto skipdirectionax
if p=5 then o=8:w=w+1
skipdirectionax
if vwpixel(w,z,poll)<>0 then w=i:z=j:o=0:gosub nextdir:p=robotAI(h):return:rem p
should get reassigned here from dir array!
return:rem 20190813 timing fix!
skipdirection2ax
if s>0 then goto skipsmoothtilemoveax
if p=1 then player1y=player1y+1:player1x=player1x-1
if p=2 then player1y=player1y+1

```



```

if p=3 then player1y=player1y+1:player1x=player1x+1
if p=4 then player1x=player1x-1
if p=5 then player1x=player1x+1
if p=6 then player1x=player1x-1:player1y=player1y-1
if p=7 then player1y=player1y-1
if p=8 then player1x=player1x+1:player1y=player1y-1
skipsmoothtilemoveax
if o=0 then p=0 else o=o-1

rem preserve player 1 x,y
rem if player1x < 8 or player1x>156 then player1y=0
rem if player1y < 7 or player1y>92 or player1x>160 then player1y=0:goto
AIskipball2
if player1y<10 or player1y>96 or player1x>160 then player1y=0:goto AIskipball2

rem --- fire bouncing shot lasting 20 frames if not in motion (var 2 dir mirrors
E)
if f=0 then f=135:score=p:missile1x=player1x:missile1y=player1y-
4:AUDC1=6:AUDF1=30
if f>127 then CXCLR=0:rem -- keep missile clear of launcher

AIskipball2

rem freeing these vars! var2=player1x:score=player1y

rem --

return
rem -----
-----
rem ----- Done AI Opponent # 2 (player1 Mirror #1) -----
-----
rem -----
-----

DoPlayer0 rem Human Player: -----

rem oops e=p:rem 20211207 automate human player with the AI for cooperative demo
play!

rem vwpixel(6,7,bindplayer1):rem player1y=70:player1x=33:COLUP1=$A8:rem was A6
rem space saver for i=0 to 1:player0colors(i)=$fe:next i: rem $B6

player0x=px:player0y=py: rem restore player 0's x,y
rem -- m = microincrement for smooth movement between tile based moves

rem --- cycle player animation
rem TOO SLOW i=m/2:i=i+b:
" lda m: lsr: sta i: bcc fastercalcdone3: inc i:fastercalcdone3 "
i=spritemap(i):loadplayer0upsidedown(i)
rem loadplayer0upsidedown(m)

if e=1 or e=4 or e=6 then REFP0=255 else REFP0=0

rem bad? if m=0 or s=1 then vwpixel(x,y,bindplayer0)

if m=6 then gosub dynamicboard

rem establish direction of tile based move if m is free
if m>0 then goto skipdirection2
vwpixel(x,y,bindplayer0): rem better?
rem trying alt loop for 30 hz split

```

```

rem if joy0fire=1 and bx=0 then:g=e

rem -- e direction :  1 2 3
rem                  4 0 5
rem                  6 7 8
i=x:j=y: rem preserve x and y for polling potential target square rollback

rem if joy0up=1 and joy0left=1 then e=1:m=8:y=y-1:x=x-1:goto skipdirection
rem if joy0up=1 and joy0right=1 then e=3:m=8:y=y-1:x=x+1:goto skipdirection
rem if joy0down=1 and joy0left=1 then x=x-1:y=y+1:e=6:m=8:goto skipdirection
rem if joy0down=1 and joy0right=1 then y=y+1:x=x+1:e=8:m=8:goto skipdirection
rem if joy0up=1 then e=2:m=8:y=y-1: rem space saving! goto skipdirection
rem if joy0down=1 then e=7:y=y+1:m=8:rem space saving! goto skipdirection
rem if joy0left=1 then e=4:m=8:x=x-1:rem space saving! goto skipdirection
rem if joy0right=1 then e=5:m=8:x=x+1

rem grab demo AI 20211207:
e=p:rem e=p got stuck in quarantine after 1m!
rem if e=0 or e=2 then e=g
rem if e=0 or e=2 then e=g else e=p
e=g
rem if e=2 then e=p

rem allow user to override for tandem controls:
if joy0left=1 then e=4:c=1
if joy0right=1 then e=5:c=2

if joy0up=0 then goto notup
e=2
if c=1 then e=1
if c=2 then e=3
notup
if joy0down=0 then goto notdown
e=7
if c=1 then e=6
if c=2 then e=8
notdown

doneusercontrol

if e=1 then m=8:y=y-1:x=x-1:goto skipdirection
if e=3 then m=8:y=y-1:x=x+1:goto skipdirection
if e=6 then x=x-1:y=y+1:m=8:goto skipdirection
if e=8 then y=y+1:x=x+1:m=8:goto skipdirection
if e=2 then m=8:y=y-1: rem space saving! goto skipdirection
if e=7 then y=y+1:m=8:rem space saving! goto skipdirection
if e=4 then m=8:x=x-1:rem space saving! goto skipdirection
if e=5 then m=8:x=x+1

skipdirection
if vwpixel(x,y,poll)<>0 then x=i:y=j:m=0:e=0:rem return
return:rem 20190813 timing fix! (trim superfluous return above)
skipdirection2
if s>0 then goto skipsmoothtilemove
if e=1 then player0y=player0y+1:player0x=player0x-1
if e=2 then player0y=player0y+1
if e=3 then player0y=player0y+1:player0x=player0x+1
if e=4 then player0x=player0x-1
if e=5 then player0x=player0x+1
if e=6 then player0x=player0x-1:player0y=player0y-1
if e=7 then player0y=player0y-1
if e=8 then player0x=player0x+1:player0y=player0y-1
skipsmoothtilemove

```

```

if m=0 then e=0:goto donesmoothtilemove
m=m-1

i=x-BITIndex: rem ----- set x,y Camera scrollbursts
if i<5 and m=1 and BITIndex>0 then q=2
if i>15 and m=1 and BITIndex<71 then q=2

j=BYTERowoffset/12:i=y-j
if i<3 and m=1 and BYTERowoffset>0 then r=2 : rem 20190808 This must scroll
vertically in increments of 2 contiguous postions - 2 and 4 vertical tile pixel
scrolls are ok, 1 and 3 break it! (why?)

if i>7 and m=1 and BYTERowoffset<196 then r=2

donesmoothtilemove rem return:rem -----

rem --- fire bouncing shot lasting 20 frames if not in motion (var 2 dir mirrors
E)
rem 20211207 autofire unless supressed
if joy0fire=0 and var1=0 then
var1=135:var2=e:missile0x=player0x:missile0y=player0y-4:AUDC0=4:AUDF0=29
rem if var1>127 then CXCLR=0:rem -- keep missile clear of launcher

rem if m=0 then
gosub movevirtualworld
rem 20190809 no variable flicker: vwpixel(px,py,bindplayer1)
rem -----
rem end gameloop subroutine, runs every frame (top vertical blank) -----
-----
rem -----
return
rem -----
rem ---subroutines -----
rem -----
movevirtualworld rem -----move the world
if q=0 and r=0 then s=0:scrollvirtualworldtoggle=0:return else
scrollvirtualworldtoggle=1: rem s=0 not supressing micro movement 20180608 queued
to scroll the CAM in any direction

rem deactivated for streamlined MBR revision of the game 20190806 , back on
s=1: rem --surpress micro movement
if q=0 then goto donehorizontalmoveCAM
rem -----
rem -- e > direction:  1 2 3 -----
rem                    4 0 5 -----
rem                    6 7 8 -----
rem -----
if BITIndex =72 then goto skipdecbitindex
if e=8 or e=3 or e=5 then BITIndex=BITIndex+1
skipdecbitindex
if BITIndex=0 then goto skipincbitindex
if e=1 or e=4 or e=6 then BITIndex=BITIndex-1
skipincbitindex

q=q-1
donehorizontalmoveCAM

if r=0 then goto doneverticalmoveCAM
if e=<4 and e>0 then BYTERowoffset=BYTERowoffset-12
if e>5 and BYTERowoffset<244 then BYTERowoffset=BYTERowoffset+12

r=r-1
doneverticalmoveCAM

rem scroll virtual world when user is in proximity to border

```

```

return:rem -----
nextdir
if h>50 then h=0 else h=h+1: rem robotAI cycle commands (BIG Trek robots!)
return

rem w=vars(0):z=vars(1):o=vars(2):p=vars(3):player1x=vars(4):player1y=vars(5)

shadowcopy rem seed i before call (varsinit autoincrements on it)
j=w:gosub varsinit: j=z:gosub varsinit: j=o:gosub varsinit: j=p:gosub varsinit
j=player1x:gosub varsinit:j=player1y:gosub varsinit
return --shadowcopy return

rem ---end subroutines -----

rem -----
rem ---gameloop2 subroutine, runs every frame:  --
rem -----

rem -- piggyback missile0 propegation for relfecting shots
rem --- propegate players bouncing shot if it is in progress:
if var1=0 then missile0y=0:goto skipshot0a
rem check for collision with the wall and bounce -- 20190829 different tones for
the bounces...
if CXM0FB >126 then CXCLR=0:AUDC0=12:AUDF0=h:gosub nextdir:var2=robotAI(h)
if var2=1 then missile0y=missile0y+1:missile0x=missile0x-1
if var2=2 then missile0y=missile0y+1
if var2=3 then missile0y=missile0y+1:missile0x=missile0x+1
if var2=4 then missile0x=missile0x-1
if var2=5 then missile0x=missile0x+1
if var2=6 then missile0x=missile0x-1:missile0y=missile0y-1
if var2=7 then missile0y=missile0y-1
if var2=8 then missile0x=missile0x+1:missile0y=missile0y-1

skipshot0
var1=var1-1

skipshot0a
rem clear collision register

rem -- e direction :  1 2 3
rem                  4 0 5
rem                  6 7 8

rem -- end piggyback missile0

rem -----

t=1-t:rem toggle

rem 20190806 deactivated for streamlined MBR revision of the game 20190806 if
t=0 then scrollvirtualworldtoggle=30:gosub DLI:scrollvirtualworldtoggle=0:return:
rem run this DLI every other frame

```

```

rem --- Character Hz throttle - show characters only if they are in camera view
if t=1 then goto DoPlayer0a
SUSTAINFORFRAMES=SUSTAINFORFRAMES+1
rem -----

rem 1st preserve existing:
i=0:gosub shadowcopy: rem this call should be in the other gameloop (balance it
if necessary)

rem mirror for sprite 1, 3rd AI player... --- automate with call wrapper (reuse
routine call above)...
rem vwpixel(20,11,bindplayer1):rem player1y=50:player1x=80:
rem loadplayer1upsidedown(0):
COLUP1=$86:rem was FA

rem pull:
w=vars(6):z=vars(7):o=vars(8):p=vars(9):player1x=vars(10):player1y=vars(11)
gosub AIOpponentTwo
i=6:gosub shadowcopy: rem preserve (push)

return:rem -----end mirror for s prie 1, 3rd AI
player
DoPlayer0a

rem mirror for sprite 0
px=player0x:py=player0y: rem hold player 0's x,y
rem 20190810

rem check for collision with AI Robot 1 and send it back home and deflect the
ball but don't destroy it:
rem need speed and space, check bit 6: if CXM0P>62 and CXM0P<127 then
CXCLR=0:v=0:AUDC0=1:AUDV0=3:bx=0:by=3:g=5:n=0:gosub nextdir:var2=robotAI(h)
rem there will be an (intentional) delay in the Grenade effect (one at a time to
allow merging concurrent fx)
rem restore player 0 mirrors x y
player0x=u:player0y=v
rem 20190815 save space here and better Fx?
if var1>122 then CXCLR=0:rem -- keep missile clear of launcher

rem 20190828 If player shoots clone play special theme (tune II)
rem BIT CXM0P: bvc dontdoreset:doreset lda #0: sta v: sta bx: sta n: sta CXCLR:
lda #6: sta AUDC0: lda #30: sta AUDF0: sta SUSTAINFORFRAMES: lda #3: sta by: sta
g: jsr Lnextdir: ldx h: lda robotAI,x : sta var2 ; : bne ballsdone :dontdoreset

" BIT CXM0P: bvc dontdoreset:doreset lda #0: sta v: sta bx: sta n: sta CXCLR:
lda #$D2 : sta COLUBK: lda #240: sta MUSICINDEX: lda #3: sta by: sta g: jsr
Lnextdir: ldx h: lda robotAI,x : sta var2 ; : bne ballsdone :dontdoreset "
" BIT CXM1P: bpl ballsdone:doreset2 lda #0: sta v: sta bx: sta n: sta CXCLR:
lda #240: sta MUSICINDEX: sta COLUBK: lda #3: sta by: sta g: jsr Lnextdir: ldx h:
lda robotAI,x : sta score ; "
"ballsdone"

rem vwpixel(bx,by,bindplayer0):rem player0x=75:player0y=20
rem superflous call, freeing cycles!!!! loadplayer0upsidedown(0):
rem space saver for i=0 to 1:player0colors(i)=$84:next i

rem -- AI, initial dir is 5 for 1st Robot , so:
data vars 3,3,0,5,0,0,4,4,0,5,0,0

```

```

data robotAI
1,8,7,2,3,4,6,5,4,2,7,1,4,6,3,8,5,2,6,7,1,4,3,2,4,1,8,7,7,2,5,5,5,4,4,4,4,2,7,1,3
,6,8,7,4,3,1,6,4,5,5,3
data spritemap 0,8,16,24,32,40,48:rem ,56 quick lookup table to save cycles
rem data fasty
0,12,24,36,48,60,72,84,96,108,120,132,144,156,168,180,192,204,216,228

rem --- cycle player animation
rem too slow: i=n/2:i=i+b
" lda n: lsr: sta i: bcc fastercalcdone: inc i:fastercalcdone "
i=spritemap(i):loadplayer0upsidedown(i)
rem i=n*8:loadplayer0upsidedown(i)

if g=1 or g=4 or g=6 then REFP0=255 else REFP0=0

rem establish direction of tile based move if m is free
if n>0 then goto skipdirection2a

vwpixel(bx,by,bindplayer0): rem binds the completed move after the micro
increments (or the initial move)

rem -----
rem -- e [g!] > direction:  1 2 3  -----
rem                          4 0 5  -----
rem                          6 7 8  -----
rem -----
rem                          6 7 8

i=bx:j=by: rem preserve x and y for polling potential target square rollback
if g=1 then n=8:by=by-1:bx=bx-1:rem space saver--- goto skipdirectiona
if g=3 then n=8:by=by-1:bx=bx+1:rem space saver---goto skipdirectiona
if g=2 then n=8:by=by-1:rem space saver---goto skipdirectiona
if g=6 then bx=bx-1:by=by+1:n=8:rem space saver---goto skipdirectiona
if g=8 then by=by+1:bx=bx+1:n=8:rem space saver---goto skipdirectiona
if g=7 then by=by+1:n=8:rem space saver---goto skipdirectiona
if g=4 then n=8:bx=bx-1:rem space saver---goto skipdirectiona
if g=5 then n=8:bx=bx+1
skipdirectiona
if vwpixel(bx,by,poll)<>0 then bx=i:by=j:n=0:gosub
nextdir:g=robotAI(h):return:rem g should get reassigned here from dir array!
goto assignandreturn:rem 20190813 timing fix!
skipdirection2a
if s>0 then goto skipsmoothtilemovea
if g=1 then player0y=player0y+1:player0x=player0x-1
if g=2 then player0y=player0y+1
if g=3 then player0y=player0y+1:player0x=player0x+1
if g=4 then player0x=player0x-1
if g=5 then player0x=player0x+1
if g=6 then player0x=player0x-1:player0y=player0y-1
if g=7 then player0y=player0y-1
if g=8 then player0x=player0x+1:player0y=player0y-1
skipsmoothtilemovea
if n=0 then g=0 else n=n-1: rem g should get reassigned from an array of AI
directions or player 1 stops after 1 move

rem preserve player 1 x,y
rem if player0x < 8 or player0x>160 then player0y=0
rem if player0y<7 or player0y>92 or player0x>155 then player0y=0
if player0y<10 or player0y>96 or player0x>160 then player0y=0

assignandreturn
u=player0x:v=player0y
rem superflous! return

rem --

```

```

return:rem not superfluous!

rem -----
rem --- end gameloop2 (bottom vertical blank)
rem -----

varsinit rem i,j indexi ,j value, then index i increments
vars(i)=j
i=i+1:return

dynamicboard
rem -- make dynamic changes to board
rem gosub nextdir
i=h+8
rem " lda i: lsr : lsr : sta j "
rem j=j+4
rem 20190828 intelligent optimization to follow green/blue robots row for the Y
tile axis (z);
rem freeing space also!
rem deprecated, too busy try this:
j=robotAI(h)+2
vwpixel(i,j,flip)
return
rem -----
rem ---KITCHENSINK subroutine, runs when scrollvirtualworldtoggle=1
rem -----
rem -----
rem -- map virtual world row colors to Camera
rem rowcolors() is a built in 1x10 array variable of Camera row colors.
rem -----
rem data xscreencolors
$b4,$b6,$b8,$c8,$88,$86,$84,$82,$fa,$f8,$f6,$f4,$38,$36,$34,$32,$1c,$1A,$18,$16
data xscreencolors
$84,$84,$84,$84,$64,$64,$64,$64,$58,$58,$58,$58,$44,$44,$44,$44,$c6,$c6,$c6,$c6
j=BYTERowoffset/12:k=j+9:l=9
for i = j to k
rowcolors(l)=xscreencolors(i)
l=l+1
if l=10 then l=0
next i: rem done mapping virtual world row colors to Camera
n=0:j=0:i=2:gosub varsinit:i=8:gosub varsinit: rem clear micro movement on
player 2,3
rem -----
rem ---end KITCHENSINK (full frame. lots of time to fit the kitchen sink)
rem -----
rem --camera pans a 20x10 tile window, 1/10th view of the gamegrid; 200 tiles
out of 2000
rem - Virtual world and sprites: a 92x20 tile semigraphics gamegrid and 8x8
pixel sprites

virtualworld
XXXXXXXXXXXXXXXXXXXXXXXXXXXXX.....XXX.XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX.XXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXX..
X.....X.....XX..X.....XX.X.....
.....XX
X.....XX.....X.....XXX.....XXXXXXXXX..X.....X.....X..X.....
....X...X.X
X..XX...XXXX.....X.....X.....XX.....XX.X.X...XXXX.....XX.X.....
.....X.X..X
X..XX.....XX.....XXX.....X...X..XX.X.....XX.X.....X.....X..X.....
.....X...X

```

```
X..X.....X.....X...X.....XX.....XXX.XXXXX.XX.....XX.X.....
.....X
X..X...X.....X..X...X.....X.....XX...X.....X.....X..X.....
.....X.....X
X..X...X.....XX...X...X.....XX...X.....X..X..X.....XX.X...XX
XXXX.X...X
X..X...X.....X..X.....XXXXX.X.....XX...X.....X..X..X.....X..X...X..
.....X.....X
X..X...X...X.....X.....XX.....X..X..X..X.....X..
X.....X.....X
X.....X.....X..XX.....XX.XXXXXXXXXX.XXX.....XX.....X.....X.....XXXXX...X..
X.....X
X..X...X.....X.....X.....X...XX.....X.....X.....X.....X.....
X.....X...X
X.....XX.XX.....X.....X.....X...XX.....XXXXX.X.XXX.....X.....
X.....X
XX..X...X.....X.....XXX..XXX..XX.....XX.....X.....X.....X.....
X.....X
X.X.X.....X.....X.....XXXXXXXXXX.....X.....
.....X.X
XXXX.....XXXXX.....X...X...XX.....XXXX.....XXXXXXXXX...X..
..XXXXXX..X
X.....XXXXXXXXXXXXXXXXX.....XXXXXXXXXXXXXXXXX.....X..
.....X
X.....XXX.....XXXX.....
.....X
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXX
```

```
player0colors $fe,$fe,100,100,100,100,100,100
rem player0colors $5c,$5c,$fe,$fe,$fe,$fe,$fe,$fe
rem running man pixel art created by PAC-MAN-RED
```

sprites

```
...X...
..XXX..
...XX..
...XXX..
...XX..
...XX..
...XX..
...XX..
...XX..
```

```
...X...
...XX..
...XX..
...XX..
...XX..
...XX..
...XX..
...XX..
```

```
..X.....
..X..X..
...XXX..
...XXXX..
...XXX..
...XX..
...XX..
...XX..
```

```
.....X..
..X...X..
..X...X..
...XX...
..X.XXXX.
```



```
..XXX..
...XX..
...XX..
```

01

```
.X...XX
.XX..X..
...XXX..
..XXXX..
...XXX..
...XX..
...XX..
.....
```

02

```
.....XX.
.XX..X..
...XXX..
.X.XX.X.
..XXXX..
...X...
...XX..
.....
```

03

```
.....Xx
.xX...x.
...XXX..
.X.XX.X.
..XXXX..
...X...
...XX..
.....
```

04

```
..X.....
..X.X...
..X..X..
...XXX..
..XXXX..
...XXX..
...XX..
...XX..
```

```
...XX...
..XXX...
...XX...
...XX...
...XX...
...XX...
...XX...
...XX..
.....
```

05

```
XXXX...
XXXXXX..
..XXX...
XXxx..X.
XXXXXXXX.
XXxx...
XX.xx...
XX..xxx.
```

```
.....
```

```
.....  
XXXXXX.  
XX...X.  
XXXXXXXX.  
XX...X.  
XX...X.  
XXXXXXXX.
```

```
.....  
.....  
..xxx..  
.x.x.x..  
XXXXXXXX.  
x.xxx.x.  
.x...x..  
..xxx..
```

chiptunes

```
4,19,4,19,8  
4,17,4,17,8  
4,15,4,15,8  
4,14,4,14,16  
4,19,4,19,8  
4,14,4,14,16  
4,19,4,19,8  
4,14,4,14,16  
4,19,4,19,8  
4,14,4,14,8  
4,14,4,14,8  
4,15,4,15,8  
4,17,4,17,8  
4,19,4,19,16  
4,26,4,26,8  
4,19,4,19,16  
4,26,4,26,8  
4,19,4,19,16  
4,26,4,26,8  
4,19,4,19,8  
4,19,4,19,8  
4,17,4,17,8  
4,15,4,15,8  
4,14,4,9,32  
0,0,0,0,48  
6,30,6,30,8  
6,27,6,27,8  
6,24,6,24,8  
6,22,6,22,8  
6,30,6,30,16  
6,22,6,22,8  
6,30,6,30,16  
6,22,6,22,8  
6,30,6,30,16  
6,22,6,22,8  
6,22,6,22,8  
6,24,6,24,8  
6,27,6,27,8  
6,30,6,30,8  
1,5,1,5,16  
6,30,6,30,8  
1,5,1,5,16  
6,30,6,30,8  
1,5,1,5,16  
6,30,6,30,8  
1,5,1,5,16  
0,0,0,0,64
```

0,0,0,0,0
7,30,7,30,24
7,24,7,24,24
0,0,0,0,0

Researching the original Vaporware product:

This collector acquisition might be the Game Loader prototype Atari Adaptor for the VIC-20! There are six switches on the unit and it attaches to the back like a sidecar similar to other adapters for the Colecovision, Intellivision and Atari's own 5200 that contained hardware equivalent to an Atari 2600 inside.

You can see the cartridge port at 8:29 and the Motorola chip with the RAM array possibly for adding fantastic Commodore graphics at 10:32. The MPU's use as early Micro Controller coprocessor is discussed at 12:35

Note the RAM Array is also 64K expanding the VIC-20's memory to match the Commodore 64:

<https://youtu.be/E0yLak9X5nw>

(Fantastic!!)
**VIC 20™ COMPUTER WILL PLAY
ATARI GAMES CARTRIDGES**
**when you plug in our
GAME LOADER!**

Wow! Now you can play all Atari game cartridges on your "VIC-20" Computer." Atari VCS cartridge video games, Activision, Imagic, M-Network cartridges will all play on your "VIC-20" Computer," when you use our new "GAME LOADER" *plus* you get fantastic VIC-20'sound and graphics.

LIST PRICE \$99.00 **SALE \$89.00**
(Includes Free ATARI Game \$32.50 List)

"15 DAY FREE TRIAL"

- **We have the lowest VIC-20 prices**
- **We have over 500 programs**
- **Visa – Mastercharge – C.O.D.**
- **We love our customers!**

**PROTECTO
ENTERPRIZES**
BOX 550, BARRINGTON, ILLINOIS 60010
Phone 312/382-5244 to order

VIC 20 is a trademark of Commodore Electronics Ltd.