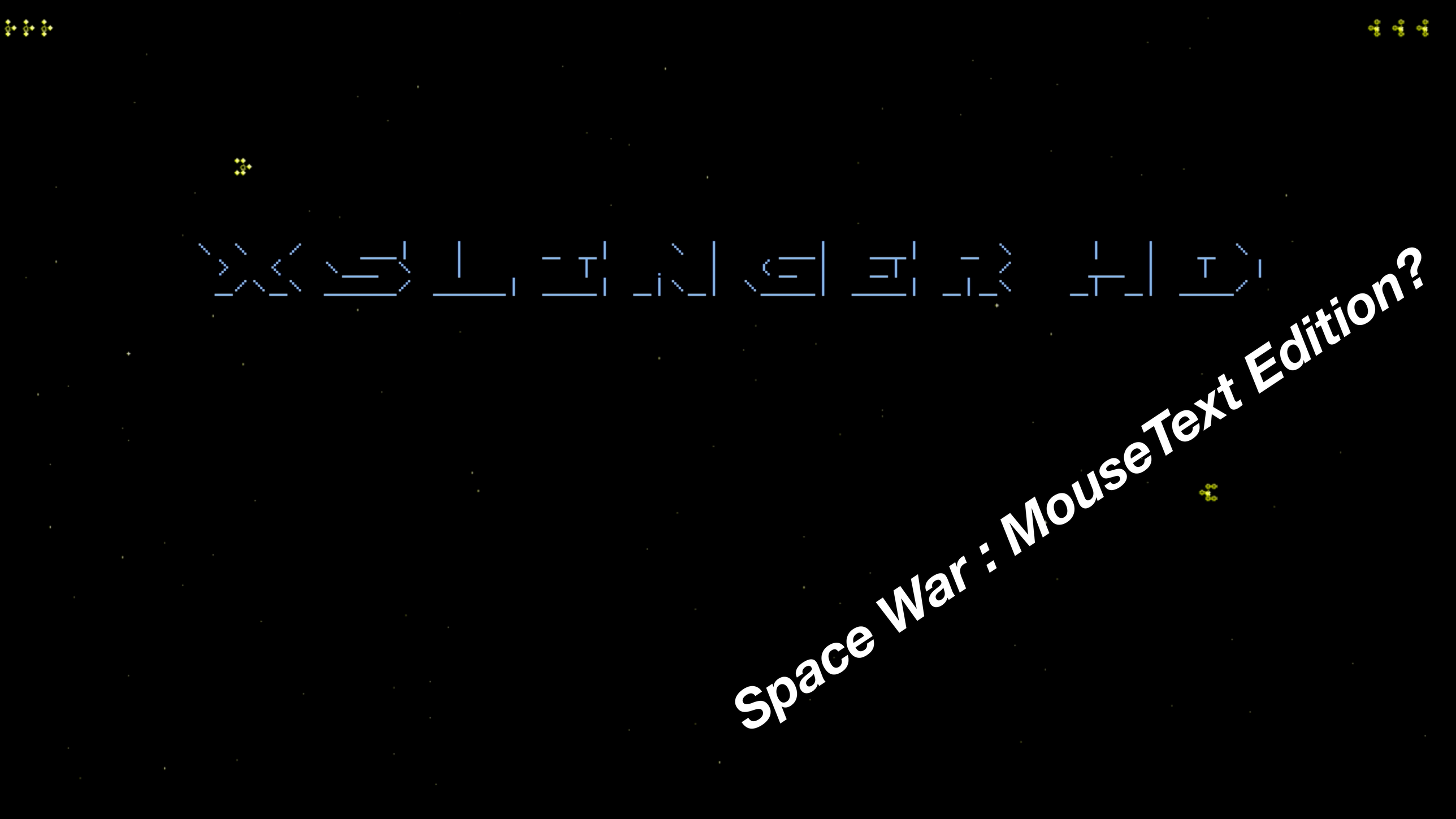
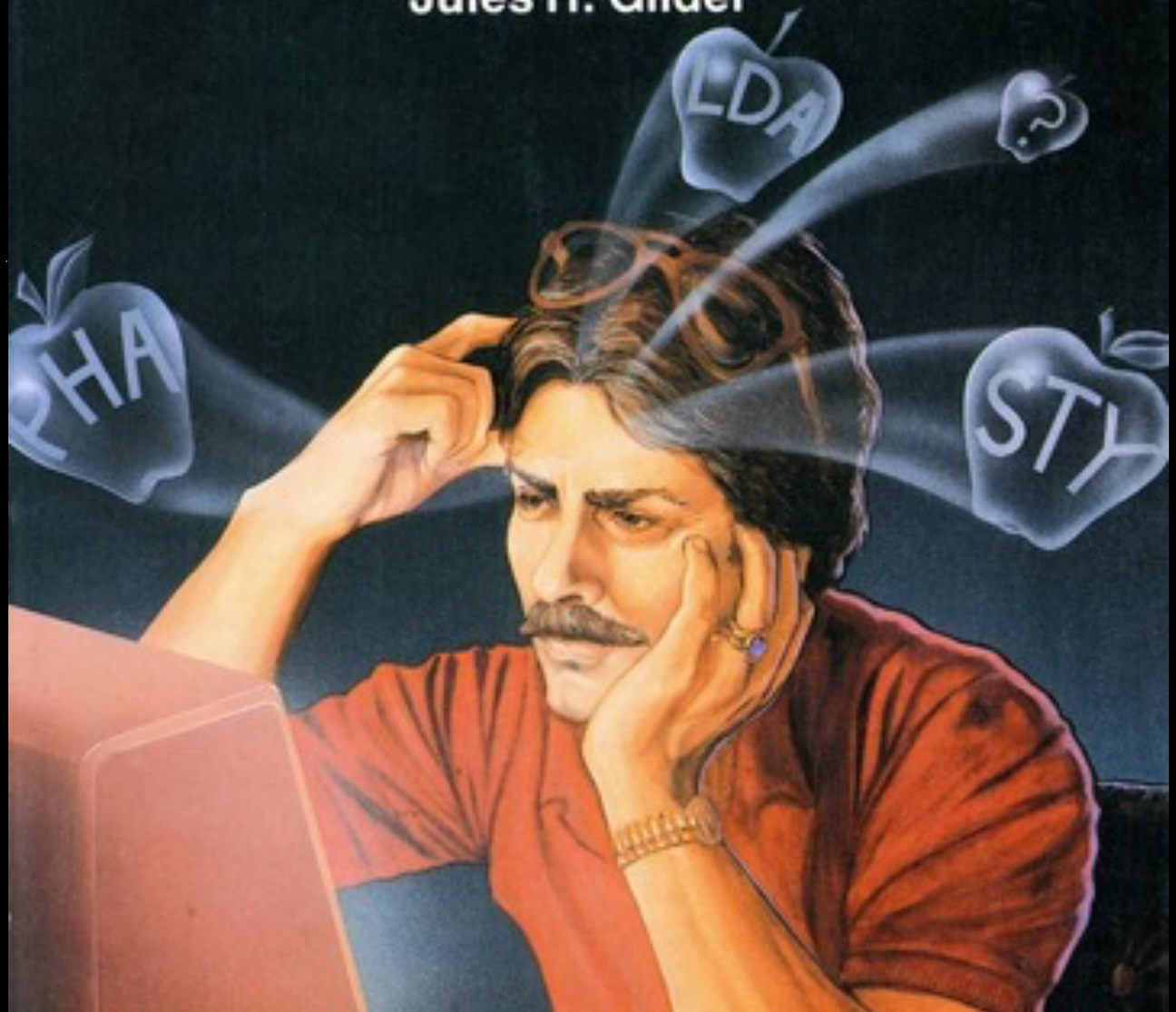
For a version of these slides including the embedded video, please visit

https://youtu.be/a7QXWan-rnE


Space War : MouseText Edition?

Now That You Know
APPLE ASSEMBLY LANGUAGE:
What Can You Do With It?

Jules H. Gilder

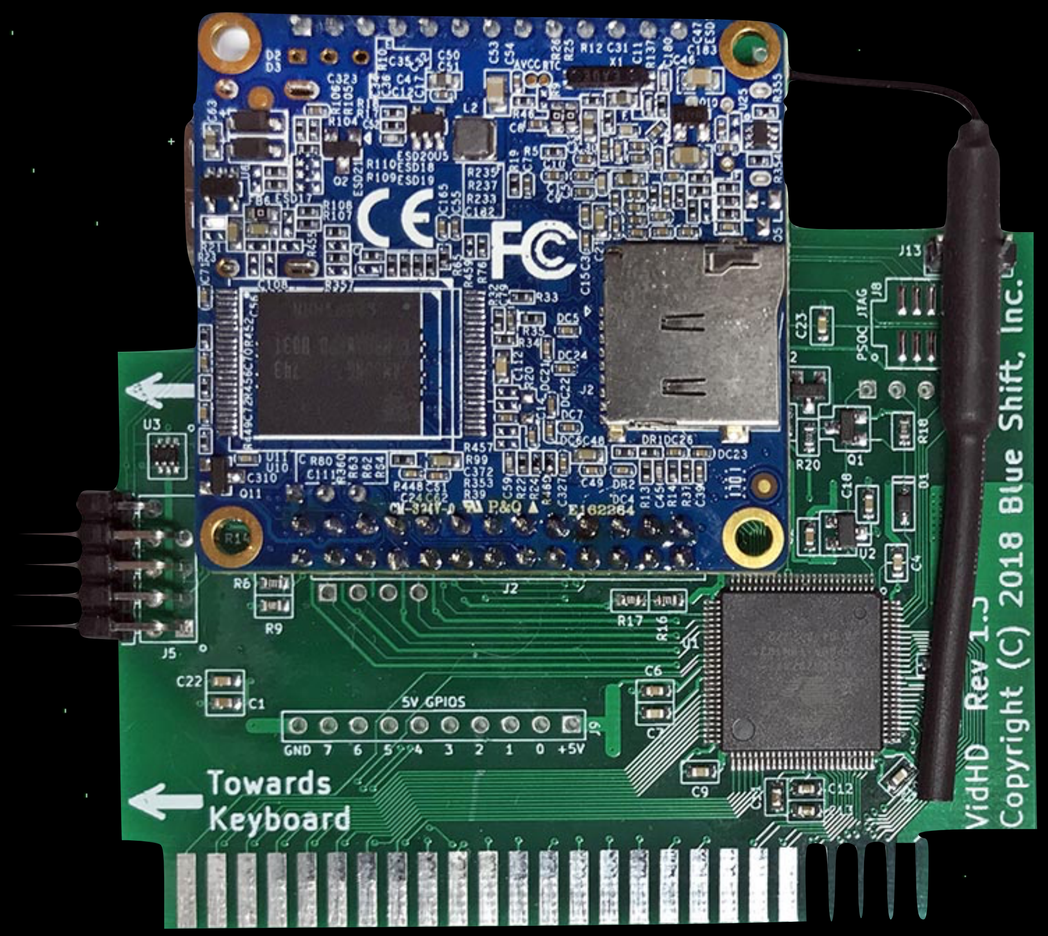


vidHD card

gives HDMI output any Apple II with slots

reproduces all standard 8/16 bit Apple II text and video modes

as an added feature, contains some new text modes that function a little differently



VidHD Text Modes

40 x 24

80 x 24

80 x 45

120 x 67

240 x 135

The Good

240 x 135

text "subpixels" are 1080p dots, on an Apple][!

scrolling

vertical scrolling (of the entire screen) is incredibly fast

plotting

can plot x,y very simply using cout or pascal txt

control

responds to all standard apple //e 80 column control characters

The Not As Good

characters

only the apple II char set (+ mousetext) is available

color

only a single color is available without scanline tricks (same as a stock Apple IIgs)

control

No way to change resolution programmatically

plot speed

(no direct access to "text RAM" on the card

actual plotting characters is quite slow - demo

plot math

240 x 135 = ~32000 characters

32000 / 12 seconds

=

~2600 characters/sec

/ 60 frames per sec

=

~43 characters/frame

```
*
00/2000:20 58 FC 9C 01 40 A9 14- X|..@).
*
00/2008:8D 03 40 A9 14 8D 04 40-..@)...@
*
00/2010:9C 08 40 20 58 FC AD 00-..@ X|-.
*
```

How to Plot

cout

(POKE 36,30)

(POKE 37,17)

```
LDA #30
STA $24    ; X

LDA #17
STA $25    ; Y
```

positioning is fast!

```
LDA #"*"
JSR $FDED
```

plotting is slow

---

pascal

```
LDA #$1E    ; GOTOXY
JSR PWRITE
LDA #62     ; X
JSR PWRITE
LDA #49     ; Y
JSR PWRITE
```

positioning is slow!

```
LDA #"*"
JSR PWRITE
```

plotting is... not as slow

```
shape_big_player1

        asc $0F,$1B,$02     inverse,mousetext on

        asc "       [       [",$02
        asc "      [[[     [[[",$02
        asc "     [[[[[   [[[[[",$02
        asc "    [[[[[[[ [[[[[[[",$02
        asc "     [[[[[   [[[[[",$02
        asc "      [[[     [[[",$02
        asc "       [       [",$02
        asc "                A",$02
        asc "               A       [",$02
        asc "             AA AA    [[[",$02
        asc "            A     A  [[[[[",$02
        asc "            A    A  [[[[[[[",$02
        asc "            A    A   [[[[[",$02
        asc "             A A  A   [[[",$02
        asc "             AA AA     [",$02
        asc $02
        asc "       [       [",$02
        asc "      [[[     [[[",$02
        asc "     [[[[[   [[[[[",$02
        asc "    [[[[[[[ [[[[[[[",$02
        asc "     [[[[[   [[[[[",$02
        asc "      [[[     [[[",$02
        asc "       [       [",$02
        asc $00
```

Plotting "sprites" is simple

Essentially moves from left to right and uses COUT at the beginning of a line or whenever a space was plotted.

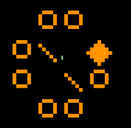
Increments vertical byte ($25) at the end of each line, and moves horizontal cursor ($24) back to the beginning.

Safety

- Surrounding COUT calls with sei/cli is necessary when using interrupt based sound/music playing

```
LDA #50     ;  X
STA $24     ;  CH cursor
LDA #77     ;  Y
STA $25     ;  CV cursor


SEI         ;  disable interrupts
JSR COUT
CLI         ;  enable interrupts
```



Safety

- A bug exists with range checking on the vertical axis.

```
coutroutine
        LDA $25     ;  CV vertical cursor position
        CMP #135    ;  134 is maximum vertical position
        BCS :dontplot
        CMP #1
        BCC :dontplot
        SEI         ;  disable interrupts
        JSR COUT
        CLI         ;  enable interrupts

:dontplot  RTS
```

You must never issue a COUT if the vertical cursor position is outside of 0 - 134 or the card will lock requiring a power off/power on boot.

NinjaTracker plus

Main Theme

Credits

Short Game
Over tunes

"popular" sounds are just played from DOC RAM ← low CPU

thrust

xslinger
fire

border
bump

larger sounds use ntpstreamsound function ← high CPU

longer

higher
sampling rate

true stereo

Jesse Blue Warning!

"mess around- shoot a powerup, & find out!"

A Few Tricks...

(Limitations are Fun!)

Colors

"3 colors are better than 1"

Step 1: set color green, Draw Player 1

Step 4: set color red, Draw Player 2

Step 2: wait for VBL

Step 5: wait for VBL2

Step 3: Erase Player 1

Step 6: Erase Player 2

Step 7: Draw everything else

<Repeat>

x + x

Colors

"3 colors are better than 1"

Step 1: set color green, Draw Player 1

Step 4: set color red, Draw Player 2

Step 2: wait for VBL

Step 5: wait for VBL2

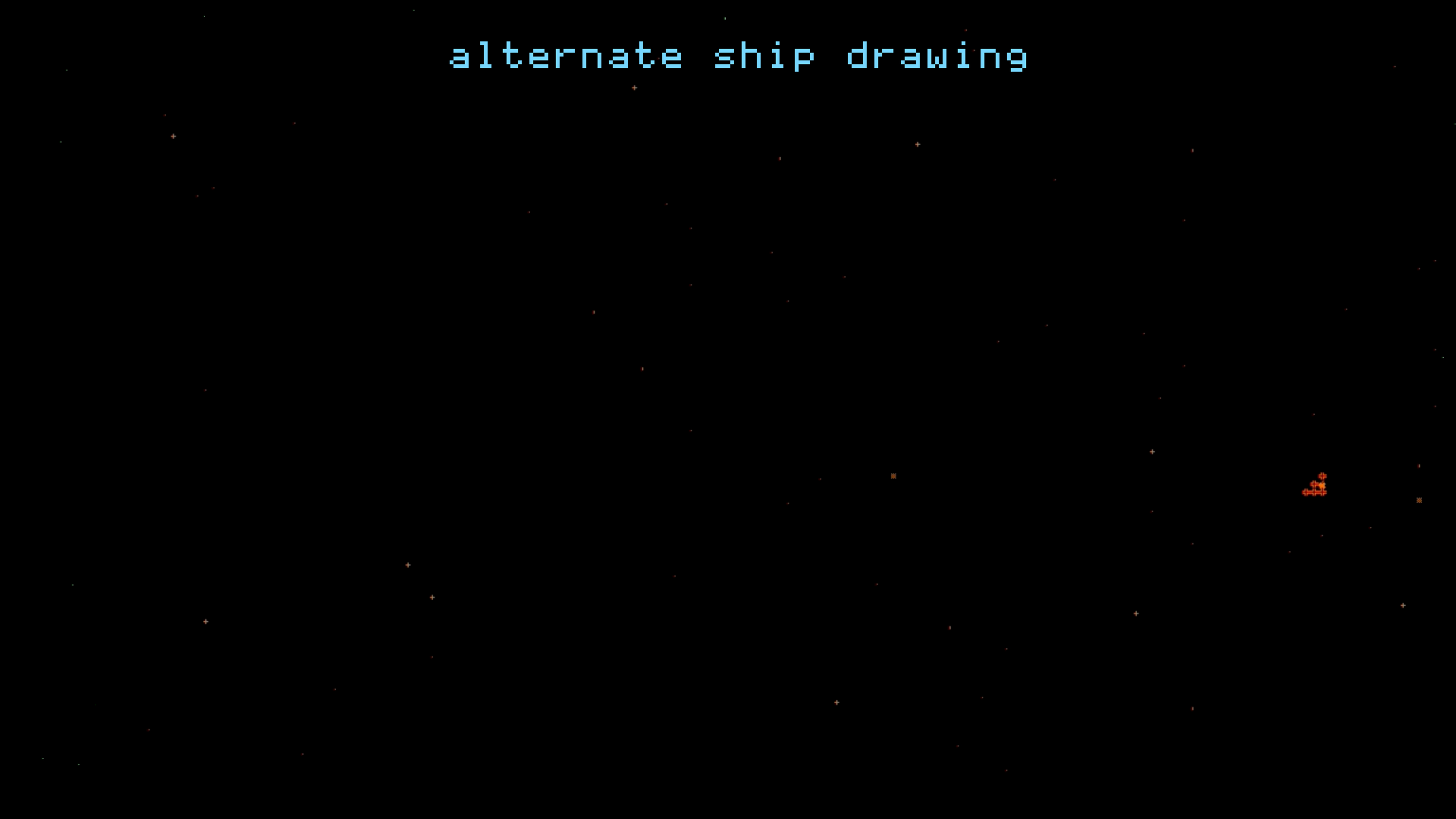
Step 3: Erase Player 1

Step 6: Erase Player 2

Step 7: Draw everything else

x + x

alternate ship drawing



Send ADB Keycode, $11

This command is used to emulate an ADB keyboard by accepting ADB keycodes from a device and then sending them to the microcontroller to be processed as keystrokes. This command does not process either reset-up or reset-down codes; these reset keycodes must be processed separately. This command can be used to detect key-up events or to emulate a keyboard with another device, such as might be used for the handicapped. This is a 2-byte command. The first byte has a value of %00010001; the second byte contains the keystroke to be processed. See the *Apple IIGS Hardware Reference* for details about the values that correspond to specific key-down, key-up sequences.

There is no current way to control vidHD text mode resolution programmatically… Right?

Idea: Use the IIgs ADB buffer to "inject" keystrokes

send ADB command byte ($11) to $c026

pause

Send ADB key-down code of the key you want

pause

Send ADB key-up code of the key you want

pause

Invoke a GETKEY prompt

Characters in the buffer fill in and perform the intended operation.

SUB	clear line	Clears the line the cursor position is on.
ESC	enable MouseText	Map inverse uppercase characters to MouseText characters.
FS	forward space	Moves cursor position one space to the right; from right edge of window, moves it to left end of line below.
GS	clear EOL	Clears from the current cursor position to the end of the line (that is, to the right edge of the window).

```
vidHD supports standard 80 column firmware control chars
```

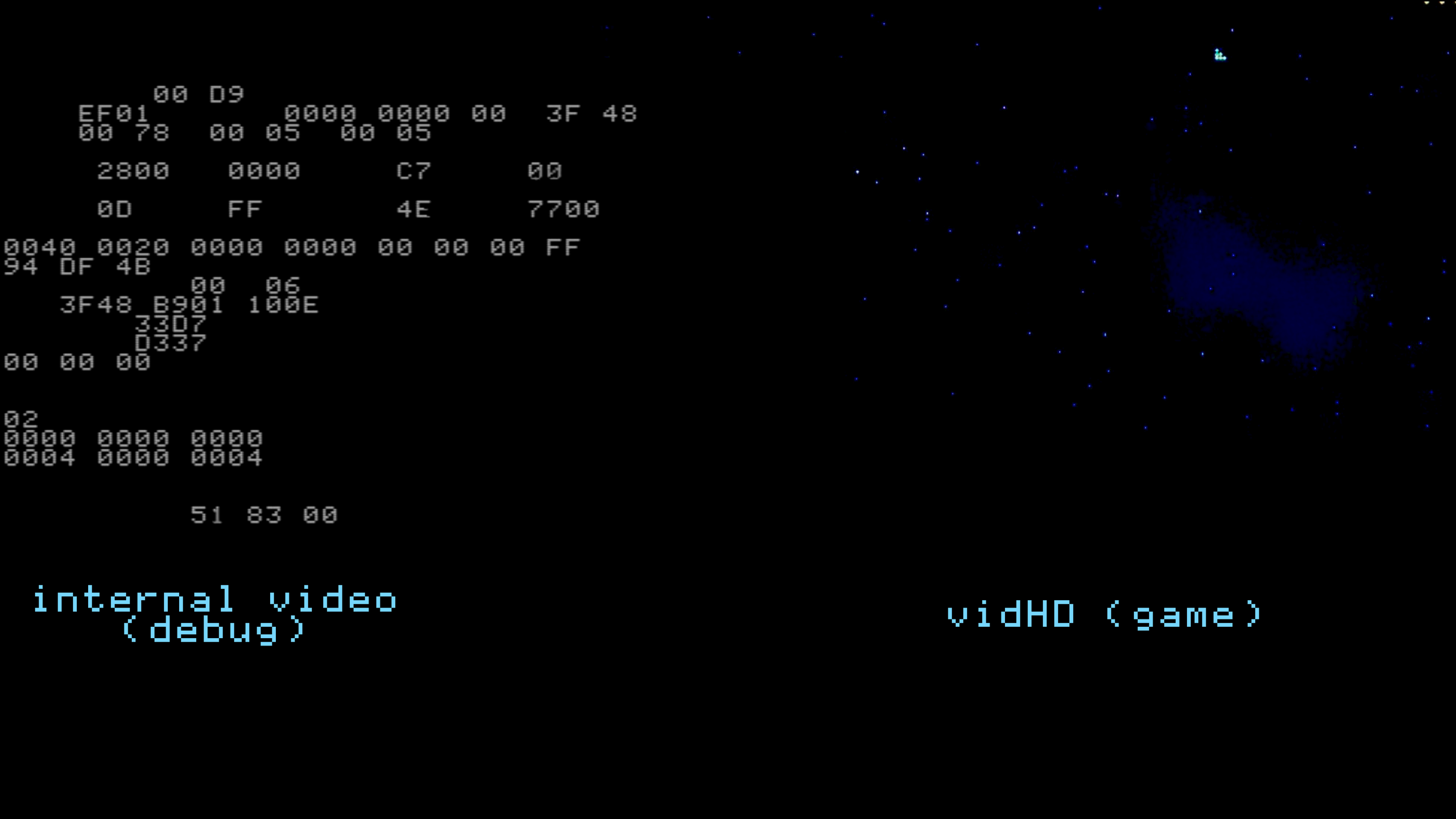
clear line

clear to end of line

* draw up to 240 characters across, fast!

* As long as your "character" is a space!

Debugging?


00 D9
EF01 0000 0000 00 3F 48
00 78 00 05 00 05
2800 0000 C7 00
0D FF 4E 7700
0040 0020 0000 0000 00 00 00 FF
94 DF 4B
00 06
3F48 B901 100E
33D7
D337
00 00 00
02
0000 0000 0000
0004 0000 0004
51 83 00
internal video
(debug)
vidHD (game)

Observations

Assembly is hard!

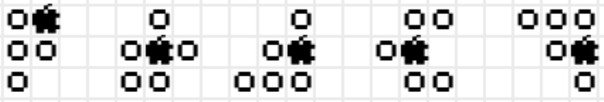
Assembly is fun!

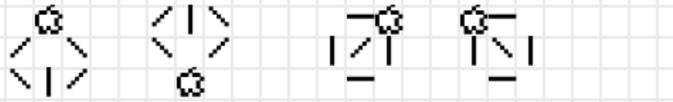
Assembly is hard!

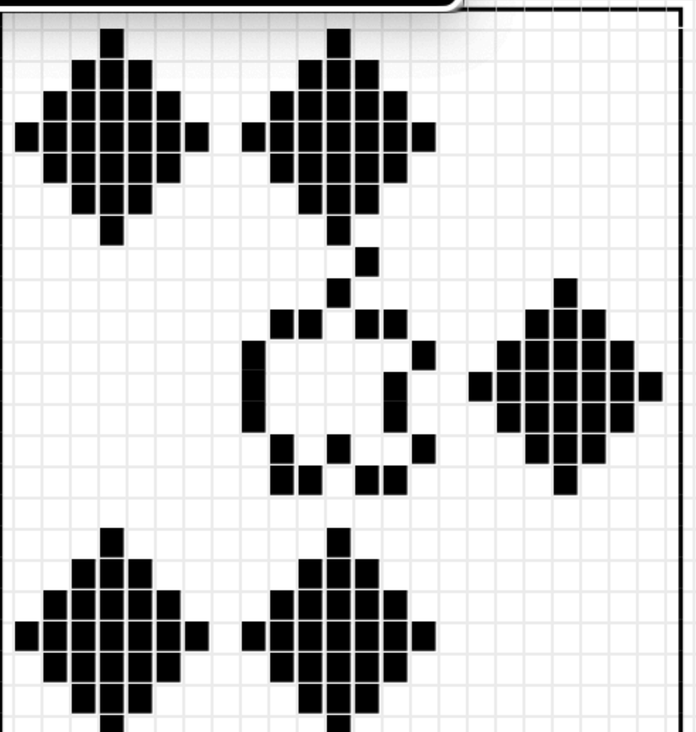
spacewar.monopic

80 100 120 140 160

MouseText







Game Demo

Zoom watchers:

See Discord comments for gameplay videos









Apple][Forever!