CULT OF 999

RISE AND FALL OF PARADROID REDUX

SUNDAY, JULY 25, 2010

Three Laws of Robotics

1. *A robot may not injure a human being or, through inaction, allow a human being to come to harm.*
2. *A robot must obey any orders given to it by human beings, except where such orders would conflict with the First Law.*
3. *A robot must protect its own existence as long as such protection does not conflict with the First or Second Law.*

It's clear that droids in Paradroid violate the first two laws, but that doesn't mean that they have to violate the third law as well. After all, Elvex dreamt of world without the first two laws.

So... why would a droid with any brains rush straight into explosion it surely knows will harm it? The most common situation where you see that happening is when a high-speed droid (like 834) shoots a low level droid in front of it and then runs (or flies, 834 is an anti-grav droid) into the explosion, destroying itself. I see no reason for that, so the droid must stop. The fastest way for detecting this was to give every patrol route segment an unique number and update droid info when it leaves a waypoint. When droid is destroyed its type is changed to explosion, but most of other attributes remain. That means that when checking for future collisions I can discard most of the droids/explosions by checking the route number only. That still leaves collisions on or near waypoints, but I have to start somewhere (and I hope I can forget more exact check later ;)

There are 433 waypoint exit directions in total so it was more efficient to write a subroutine to enumerate all routes on fly than including them in the data. That routine is less than 130 bytes long, static data plus depacking code would be at least twice the size. I also needed one 256-byte table to be able to look up the route number fast; 32 waypoints with 8 possible directions form an 8-bit index into that table. I can update route number with "lda waypoint_num; asl; asl; asl; ora dir; tay; lda routes,y; sta droidRoute,x" (that happens only when droid leaves waypoint) and check for impending collisions with "lda droidRoute,x; cmp droidRoute,y" - only if routes match I need to check for the distance between two objects. Nice and fast, now I need to play some games to check if I can see the difference.

Later I can use the same data to check if another droid is in a security droid's way, and if that's the case the security droids may decide to destroy a low-level droid to serve a greater good - to protect the ship.

A robot will guard its own existence with lethal antipersonnel weaponry, because a robot is bloody expensive.
- David Langford

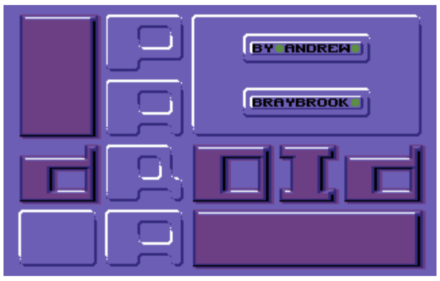
POSTED BY TNT AT 1:21 PM 3 COMMENTS:

SUNDAY, JULY 18, 2010

Not dead yet, part 2

It may have taken two years, but here is the newest and greatest version!

Both original and Metal Edition graphics are now in the same executable, droids have slightly modified AI, high score saving should work with most common expansions (you may have to disable your disk speeder



cartridge though) and there is a new frontend. Can you name all the games from where I borrowed something?

Some minor changes including but not limited to

- orange/red alert increases enemy fire probability half/full ship
- competition mode - same droids every time, has separate high score file
- no pacifist bonus if disruptor used
- droid centered on lift when deck changes - no more exit through wall

File update: two bugfixes to eliminate crash on startup (hopefully one of them does the trick), one minor visual fix and adding disruptor to firing statistics. No more easy accuracy bonus by using 711/742 only!

-

POSTED BY TNT AT 11:46 AM 4 COMMENTS:

FRIDAY, JANUARY 1, 2010

Crash Boom Bang!

Ever tried disrupting the last droid on ship to smithereens and entering lift when the explosion was still going on? Not? Good. That would have crashed the game right there. Very annoying, especially if you had just quadrupled your all time high score...

What are the changes of someone completing a ship and entering lift before deck is shut down? I don't know, but I managed to do so! Then I reproduced it on purpose to make sure there is a bug. Now I have to hunt it down.

Edit: Found it! It's the same bug which causes "pacifist takeover" (not shooting any droids - except with disruptor!) crash randomly at the same point. In Time is of the Essence I gave you the code for play are top split. In Irq_118() I use this to stabilize raster regardless of how many sprites are over the split:

```
     lda     $dc04           ; [1,15] ([2,15] if NTSC/Drean)
     eor     #$0f            ; [14,0] ([13,0])
     sta     .j3+1
.j3     bpl     *+2             ; jump into the delay code

;       entering at offset 0 delays 16 cycles,
;       entering at offset 14 delays 2 cycles
;
;       OP_CMP_IMM is opcode for CMP #immediate (2 cycles),
;       OP_CMP_ZP is opcode for COM $zeropage (3 cycles)

     cmp     #OP_CMP_IMM
     cmp     #OP_CMP_IMM
     cmp     #OP_CMP_IMM
     cmp     #OP_CMP_IMM
     cmp     #OP_CMP_IMM
     cmp     #OP_CMP_IMM
     cmp     #OP_CMP_ZP
     nop
```

Guess what happens if CIA timer underflows before the interrupt code is executed? It will jump randomly forward and CPU ends in la-la land. Unfortunately there is some code which needs to be run with interrupts disabled, and if top split is delayed because of it... BOOM!

The solution? I can use software semaphore to protect shared resources and avoif disabling interrupts. Nice and clean solution, but why should I do that when I can kludge around the problem by waiting for raster position after the split before disabling IRQ... "bit $d012; bpl *-3; sei" fixes the bug.

POSTED BY TNT AT 6:12 PM 6 COMMENTS:

WEDNESDAY, DECEMBER 30, 2009

NTSC Blues

Why didn't anyone remind me that it would be nice if the new front-end worked on NTSC as well...

Answers on a postcard, please.

POSTED BY TNT AT 8:14 PM NO COMMENTS:

SUNDAY, JULY 6, 2008

Time is of the Essence

After reading about Mike Dailly's raster split trouble I decided to write a bit about how the top split is done in Paradroid Redux.

There are three things to do when playing area begins:

- change background color
- start character display
- start sprite display

The first one is just $d021 change, other two require a bit of trickery.

To mask vertically scrolling characters one would usually use illegal graphics mode (Extended Color Mode comined with Bit Map Mode and/or Multi Color Mode) or sprites. The first method produces black pixels so it isn't usable unless background is black, and the second one is unusable if sprites need to cross the split.

So, it's time for some font trickery. Raster interrupt several lines above the split changes to blank font, and then the actual split interrupt changes $D018 to display correct character data. This means "wasting" $0800 bytes for the blank font, but half of that memory is used as temporary buffer elsewhere so actual cost of clean split is one kilobyte.

Clipping sprites cleanly requires trickery as well as there is no way to start sprite display from the middle of graphic data. One way to achieve clipping is clearing top of sprite graphics if it overflows top split, but that would waste time both when clearing the memory and when running extra animator/digit generator rounds to restore top of sprite when more of it gets shown. Another way to achieve clipping is to put sprite at non-visible x-coordinate and then change them at the correct line. However, there is no time to change multiple registers in time.

Guess what? $D018 comes to rescue once again. There is an extra screen where the sprite pointers point to blank sprites. When the time comes, split interrupt doesn't only change font (bits 1-3 of $D018) but also displayed screen (bits 4-7 of $D018). Nice and easy solution but it requires a blank screen, another one kilobyte wasted. No, not really - there is no reason why one half of the blank font couldn't be used for blank screen. What that means is that sprite clipping is practically free as there is no extra memory required and $D018 needs to be written anyway for character blanking.

There is one problem with the screen change though. While VIC-II reads font data and sprite pointers & sprite data every line, character pointers (screen data) are only read on every eighth line. This means that while sprite and font changes are immediate, screen change affects display 0-7 lines later. To overcome this the top line of playing area is copied onto blank screen so character pointers are correct when $D018 is changed.

As blank screen is located inside blank font this copying creates yet another problem. Blank font isn't that blank any more. The topmost line is at SCREEN + $140, which means that chars $28-$2c aren't blank and will produce garbage if they appear on the topmost line. The easiest way to avoid that is to not use those chars inside playing area at all, so that's what is done. The same problem happens because of blank sprite pointers at SCREEN + $3F8, char $7F. That one is unused as well.

And what does all this has to do with timing problems Mike mentioned?

VIC-II doesn't have time to fetch sprite data when CPU is not using memory bus, so it has to stop CPU momentarily whenever sprites are active. Just how many cycles VIC-II steals from CPU depends on which sprites are active, and this causes a timing hell as you have to change registers when C64 is inside the side border area to avoid flicker. With $D021 change you have all the side border (23 cycles) to change it, but $D018 is trickier. Sprite data is fetched very early, the first three sprites get their data read at the end of previous line. This means that if $D018 write is late sprites will stay blank for one extra line.

To get register writes done at the very beginning of side border area the game uses CIA timer to stabilize raster timing. During game init CIA1 timer A is started, running through 63-65 cycles depending on VIC-II version. This means that $DC04 is always synchronized with current display X position. IRQ only needs to read $DC04 and skip that many cycles.

Did I forget bad lines? Every eight line VIC-II needs to read character pointers and that's not possible without stopping CPU for most of the raster line. This means that there is absolutely no time for anything unnecessary. In the case the split happens on a bad line the game triggers raster IRQ two lines above split, prepares next interrupt at the correct line and then preloads $D018/$D021 values and executes two-cycle instructions until IRQ happens. That guarantees minimum interrupt latency. When raster interrupt happens it will just write those two registers, clean up the stack (this second interrupt pushed status register and return address into stack) and jumps into the common code.

Nothing explains code better than source, so here it is. Only relevant parts are shown, and for clarity I've removed all assembly directives which were there to make sure branches don't span page boundaries (which would add one cycle to the branch).

```
       IRQ at line 95, prepare for split

        ...

        lda     #$10
        ora     _vScroll
        sta     $d011
        cmp     #$16
        bne     .057

; special case for bad line

        lda     #<Irq_116
        sta     $fffe

        lda     _d018+1
        sta     _d018b+1
        lda     _d021+1
        sta     _d021b+1
        lda     #116
        bne     .x1             ; jmp

; normal case

.057    lda     #<Irq_118
        sta     $fffe
        lda     #118

.x1     sta     $d012

        ...

;-----------------------------------------------------------------

;       this one used when ($d011 & 7) = 6, stuffs
;       d018/d021 as fast as possible at raster 118

        subroutine
Irq_116 pha
        sty     .yr+1
        cld

        lda     #<Irq_118b
        sta     $fffe
        lda     #>Irq_118b
        sta     $ffff
        lda     #118
        sta     $d012
        inc     $d019

;       118/15 = 7.8 so this one is executed 8 times

        sbc     #15
        bcs     *-2             ; 8*5-1=39 cycles

;       preload registers and execute 2-cycle
;       instuctions until next IRQ happens

_d018b  lda     #scr_GAME
_d021b  ldy     #0
        cli
 repeat 16
        cli                     ; 32 cycles wasted
 repend

;       now is the time to write registers, we always enter
;       via interrupt as the above code never runs this far

Irq_118b
        sta     $d018
        sty     $d021

;       clean up stack and continue normal IRQ code

        pla                     ; flags
        pla                     ; PC lo
        pla                     ; PC hi, always != 0
        bne     .irq0           ; jmp

;----------------------------------------------------------------

;       normal case, use timer value to stabilize
;       raster regardless of sprites over the split

Irq_118
        pha
        sty     .yr+1
        cld

        lda     $dc04           ; [1,15] ([2,15] if NTSC/Drean)
        eor     #$0f            ; [14,0] ([13,0])
        sta     .j3+1
.j3     bpl     *+2             ; jump into the delay code

;       entering at offset 0 delays 16 cycles,
;       entering at offset 14 delays 2 cycles
;
;       OP_CMP_IMM is opcode for CMP #immediate (2 cycles),
;       OP_CMP_ZP is opcode for COM $zeropage (3 cycles)

        cmp     #OP_CMP_IMM
        cmp     #OP_CMP_IMM
        cmp     #OP_CMP_IMM
        cmp     #OP_CMP_IMM
        cmp     #OP_CMP_IMM
        cmp     #OP_CMP_IMM
        cmp     #OP_CMP_ZP
        nop

_d021   ldy     #0
_d018   lda     #scr_GAME
        sta     $d018
        sty     $d021

;       continue with interrupt
.irq0   ...
```

POSTED BY TNT AT 7:17 PM 8 COMMENTS:

MONDAY, JUNE 23, 2008

Have a nice summer solstice(ish)

To honor the sun I give you a new release - or two releases actually.

My plan was to build an unified version with both graphic sets held in memory all the time, but that didn't happen due lack of memory. Too bad, having different graphics on different decks would have given some more variety to the game.

I **did** manage to fit all necessary data into memory, but as I had to
EOR fonts together there was no way to swap between them without temporary 2 KB buffer. You really can't unpack LZ data and EOR it simultaneously, which took me way too long to realize.

While it would have been possible to do the EOR in two passes with the 1 KB buffer I do have, I didn't bother with that as I would have to drop dual graphics as soon as I need the memory back anyway.

Oh, I did fix one single pixel bug in the hires font too :)

Edit: I also added single pixel bug into Metal Edition - when you clear deck the first time, there may be extra pixel in the background star. That one is gone as soon as you move a bit vertically, so I won't do another build just to fix it.

POSTED BY TNT AT 11:43 PM 1 COMMENT:

SUNDAY, APRIL 20, 2008

Not quite dead yet

Here is little something for you who fear that the project is dead. Not much have changed though:

- Subgame should now take 10 seconds regardless whether you're playing on C64 or C128. That's still slightly longer than original, but way better than 12 seconds.
- Game should finally be Drean 64 compatible. I thought I put the code for this into the game 18 months ago, but apparently I didn't. Well, who's going to send me a Drean 64/128 so I can actually test it?
- Most likely there are some other changes too, it was six weeks ago when I last touched the source. The only reason I did it now was to fix the download link.

Note: if you for some reason want to archive every single release, then do yourself a favor. **Don't use build number as filename part!** It was never meant for that. Instead, parse it as BB-DDMMYY where BB is daily build count, DD is day of month, MM is month and YY is year. Then reorder these as YYMMDDBB and when using that as part of filename you get chronologically sorted list.

Edit: Drean compatibility is now confirmed, thanks to the_woz. Check out his blog, especially Drean-specific entries.

POSTED BY TNT AT 10:51 PM 5 COMMENTS:

TUESDAY, DECEMBER 25, 2007

Have a nice winter solstice

For those too busy to read any further, click here.

Important: archive updated January 2nd, you need to delete old high score file as it's not compatible any more.

Due to some rather unfortunate events in the family I haven't had as much time for PR as I would have liked to, so there are no major changes. Minor changes include:

- fixed all but one of known bugs.
- 2500 bonus points if you clear a ship without shooting any enemy droids. Note that if you hit a single droid on ship one, you won't get bonus even if you clear ship two without any shooting as hit counter is preserved from one ship to the next (same is done with accuracy calculation).
- you can reduce enemy droid pulser count in the subgame by damaging them. This doesn't have much effect with the higher class droids, and you will have to cope with whatever energy the droid has left...
- background stars are a bit more interesting now.
- as always, it's slightly smaller and faster :)

As I haven't had much time to test this one, report any oddities please.

Changes which didn't make it to this version:

- subgame bonus points for 11-1 / 12-0 wins. No time to fix the bugs caused by this...
- raiders. You know, those annoying rogue droids in Paradroid'90.

Even if my time for coding has been limited, that doesn't mean that I haven't thought about the game during the slow times at work. I'm positive that the actual playing area can be enlarged by t least one character row. With C128 I think it might be possible to do two or three additional rows without the game slowing down. We'll see if I ever have time for that.

POSTED BY TNT AT 8:21 PM 18 COMMENTS:

SUNDAY, OCTOBER 7, 2007

I'm sane, thank you :P

Contrary to some other claims I'm not crazy, or at least I imagine so.

That's not the only error in Paradroid talk page - so here we have (drumroll, please)

The Definite C64 Paradroid Version Guide

- Paradroid (original), 1985

- Paradroid Competition Edition, 1986
  This one is identical to the original, except that it has some vertical blank waits removed. That allows the game run faster most of the time.
  Scroll code is unchanged, whoever wrote that it was enhanced clearly hasn't disassembled all versions and done comparisons between them...

- Paradroid Metal Edition a.k.a. Heavy Metal Paradroid, 1986
  Minor changes, mostly allowing the use of multicolor chars, remaining ones save couple of cycles and/or bytes here and there.
  Uses C128 2 MHz mode in top/bottom border for higher speed.
  Fixes the decimal mode flag bug which causes weird sound fx in earlier versions.
  Some scroll text changes - this includes two bad chars which seem to be in every original ME tape!

- Paradroid Redux, 2006-
  Nanos gigantium humeris insidentes.

POSTED BY TNT AT 12:17 AM 6 COMMENTS:

MONDAY, SEPTEMBER 24, 2007

Tweaking

Too little time for anything major (yet!) but scoring and subgame have seen some little changes.

- Bonus score if you do well in subgame. 20% bonus for each remaining pulser if you win 11-1, 40% if you win 12-0
- 2000 point bonus if you're wimp and use only transfer to overtake the ship. Not enough to compensate for score lost by not shooting droids, as I don't want to encourage that kind of cowardism ;)
- Shoot droids to pieces before transferring to them - their pulser count reduces by one for every 16 points of damage. Don't forget that you have to cope with whatever energy they have left, though!

In addition to adding small things I've also discarded some ideas

- Grenades/mines. What to do when droid explodes a mine but there are no free sprites?
- Two player mode through link cable. That cuts down sprites available for enemy droids and fire, lowering the difficulty considerably. With fever sprites it's also harder to hide the fact that the game teleports droids away if it runs out sprites.

I have doubts about transport pads as well. These would transfer player within a deck, but that would require resetting droid positions to avoid several visibility problems. And that would mean teleporting all droids, meaning you could face the same robot you were running away just a second or two ago half a deck away. You may say that there isn't much realism in the game, but that makes it even more important to preserve what's left of it!

POSTED BY TNT AT 7:24 PM 2 COMMENTS:

WEDNESDAY, SEPTEMBER 5, 2007

Wasting time

How can one waste gazillions of cycles to mirror one sprite? Quite easily, just forget speed and concentrate on compact code.

```
MirrorSprite

    ldy  #0
    sty  src
    sty  .5+1

;   src = A<<6 | $4000

    sec
    ror
    ror  src
    lsr
    ror  src
    sta  src+1

;   ptr = X<<6 | $4000

    txa
    sec
    ror
    ror  .5+1
    lsr
    ror  .5+1
    sta  .5+2

;   get sprite multicolor flag

    ldy  #$3F
    lda  (src),y
    sta  tmp2      ; b7=1 if multicolor

    lda  #60

.1  tax            ; x=60,61,62, 57,58,59, ... 3,4,5, 0,1,2

    lda  #3
    sta  bytesLeft

.2  dey            ; y=62,61,60, 59,58,57, ... 5,4,3, 2,1,0
    lda  (src),y
    sta  tmp1
    lda  #$01
.3  lsr  tmp1      ; 5
    bit  tmp2      ; 8
    bpl  .4        ;10  hires, 8 loops 1 bit each

    php            ;13  else multicolor, 4 loops 2 bits each
    lsr  tmp1      ;18
    rol            ;20
    plp            ;24
.4  rol            ;26
    bcc  .3        ;29  8*16=128 / 4*29=116
.5  sta  $8000,x   ;    63*128=8064

    inx
    dec  bytesLeft
    bne  .2

    txa
;   sec            ; asserted with "bcc .3"
    sbc  #6
    bpl  .1

    rts
```

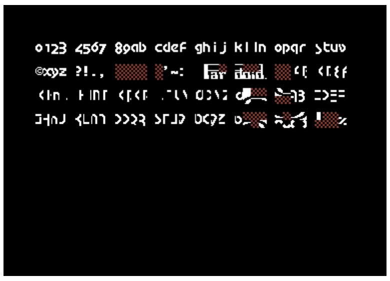
Hey, I just realized I can make it at least one byte sorter! ;)

Update: cycle counts were way off...

POSTED BY TNT AT 1:44 PM NO COMMENTS:

TUESDAY, SEPTEMBER 4, 2007

Byte Liberation Front



I keep surprising myself with all the memory I can squeeze out when I start trying. When reordering the multicolor charset from the Metal Edition Paradroid I finally took a good peek at the alpha charset. Eliminating duplicate chars and blanks freed another 31*8 bytes and couple more in graphics font. There are several 8-byte data areas I can scatter anywhere in the memory, but to keep it neater I should do some rearranging. Finding the correct data block in binary file whenever I want to change something doesn't sound fun to me.

In case you're wondering about those five blank chars below "8" - "c" and why they aren't used... Chars $28-$2c (top halves of those 1*2 characters) are at offset $0140 and need 40 bytes. (Not so) interestingly that's the same offset and size as 10th character line on screen, which is the first line used for the deck display. To change all sprite pointers and displayed charset with single $d018 write I simply switch from blank screen + blank font to actual game screen and graphics at the correct line. However, VIC-II has already read the character pointers at that time (from blank screen) which means that I have to copy the top line from game screen into the blank screen. That in turn makes chars $28-$2c visible if used because the blank screen is actually the first half of the blank font... So, those chars are unusable as graphic data both in alpha font and in in-game graphics font. That however doesn't mean that they can't be used for anything else ;)

From graphics to sound effects... I found out that I can move the main SFX table partially into the RAM under I/O at a cost of ten cycles every time an effect is started. This filled up the hard-to-use $D000 RAM completely, and freed 160+ bytes elsewhere.

Back to graphics with something which is only an idea so far. Compressing droid parts separately would gain ~600 bytes, but I can do even better. If sprites are decompressed into one continuous block, I can use the previous sprite data as codebook which would give better compression at no extra cost. Extra time used for decompression isn't anything to worry about, no one has complained about the sprite mirroring delay yet and that one takes about 15,000 cycles per sprite, 60,000 cycles in total - even if mirrored sprites aren't used at all!

All these summed up give me more than one kilobyte to play with - but what should I fill it with?

POSTED BY TNT AT 8:34 PM 1 COMMENT:

SUNDAY, AUGUST 26, 2007

Lies, damned lies, and statistics

I needed something to use when deciding which tables benefit most from move to zero page, so I counted words and their occurences in all source files. In addition to giving me the info I wanted, it revealed something else.

Paradroid Redux Top 30 ML Instructions

```
  1549    lda            202    adc            119    bmi
  1362    sta            175    lsr            108    iny
   679    jsr            167    bcc            106    clc
   394    ldx            163    stx            100    sbc
   380    ldy            151    bpl             85    dec
   313    bne            144    asl             84    eor
   251    rts            133    bcs             83    ora
   245    beq            131    inc             82    bit
   219    cmp            130    sty             81    sec
   210    and            124    jmp             76    dex
```

Disclaimer: Above counts include some code which is commented out, and I didn't expand all macros either.

POSTED BY TNT AT 1:53 PM NO COMMENTS:

SATURDAY, AUGUST 25, 2007

Zero page roundup

I finally took the time to change all zero page variable declarations from

```
var = $02
```

to

```
var ds.b 1
```

and removed all variables which aren't used any more. I suspected I would end up with 30 to 40 free zero page locations, but it turned out to be about 70 bytes! I swiftly used half of it for the two most used object variable tables (16 bytes each), which makes accessing them both faster and smaller. I guess those tables really are accessed a lot, code size was reduced by over 100 bytes... I still have room for another two tables there, but I have to find the ones which gain most cycles/bytes.

Complete list of bug fixes since the previous release:

- Droid teleportation doesn't send them outside the deck any more.
- Waypoint chars are placed onto deck map every time when exiting lift, meaning no more completely confuzed droid army.
- Door status is restored when exiting lift, so droids won't get stuck against them any more.
- The first deck entered after load had waypoint magic chars visible, as they aren't hidden in EnterShip() routine but in EnterDeck(). Instead of wasting three bytes to add one JSR call, I changed the font so magic chars are there after load :)
- Background stars are now really disabled during the intro sequence, that avoids random colors for lower half of "7" char. I disabled them earlier, but that broke when I added more run-time randomizing.

Important bits here.

Update

Two more bugs fixed, one remains... Intro text scrolled beyond the end of page sometimes, and out-of-sight droids were paralyzed on C128. The remaining bug allows droid go through wall in certain conditions, something which I didn't notice on emulator.

POSTED BY TNT AT 8:59 PM 3 COMMENTS:

FRIDAY, AUGUST 24, 2007

Squish 'em

Two bugs found and fixed, both of them having the same root cause. Whenever you change deck in lift, it's built immediately, even if you don't actually enter that deck. This avoids a delay when you exit the lift. However, if you then decide to go back to the deck where you entered lift after, game keeps previous droid/deck status intact but deck map is reset to the default state.

Now, the two bugs:

1. All waypoints are now marked with magic chars for faster detection, and I didn't restore them when re-entering deck. Mea culpa. Now waypoints are decompressed/marked correctly evrey time. This

means that some work is done twice when entering a new deck, but it's fast enough to be unnoticeable.

2. Doors keep their state on re-entry, but deck map has all doors closed after decompression. If droid had opened a door when you caused level build in the lift, droid got stuck against/in the middle of a door which was visually closed, but logically open. This bug was there from the beginning. I added some code to restore door visual state when entering a deck, and it seems to work. Doing this with minimum amount of new code made door code resemble spaghetti tho.

New beta out during this weekend, with almost all known bugs fixed. Well, make that all known bugs fixed if I have time for it :)

POSTED BY TNT AT 12:36 PM NO COMMENTS:

SUNDAY, AUGUST 19, 2007

Another week gone by

And so time passed by with nothing extraordinarily great done...

I found a bug in the latest beta - it teleports droids to the great unknown because I didn't adjust the teleport routine when I changed the waypoint system. Fixing that didn't take long, but with the released beta you need to enter another deck and come back to get droids back. Just visiting a lift doesn't work.

Multicolor is almost working. Almost, but not quite. I **really** should have done multicolor changes too when I changed the hires font...

I saved 70 bytes in the background star drawing by combining the top and bottom routine, it's not time critical so I could shrink the code without worrying about the speed. Adding random stars and blinking took most of the freed space, but I guess I can accept that.

I've used some of the new droid-specific data in old routines, speeding them up slightly as well as making them smaller. Inserting new droid AI makes every droid take slightly longer time to run, so I need every cycle I can get. I want to make big bad droids get irritated by the smaller ones when alarm is high if possible. I can already see 834 bumping into 329, shooting it out of the way and then rushing into the explosion. :)

One thing to consider is to make two passes through the droids when running them, handling visible ones in the first round and the remaining ones in the second. That way I could exit early if it looks like I'm running out of time. Sorting out-of-view droids by their distance would make it work even better, but sorting takes time...

POSTED BY TNT AT 8:05 PM NO COMMENTS:

MONDAY, AUGUST 13, 2007

How about a nice game of chess?

No chess here, I'm afraid. However, if you prefer more action then try the latest version of Paradroid Redux instead. Every bug listed here has been fixed (I hope).

I guess I now can go back to do multicolor changes. I really should have reordered the MC font when I did the hires one, now I need to dig out old notes telling which char ended where.

Update

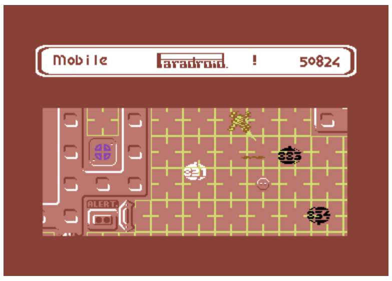
It seems that this version finally runs solid 25 Hz on PAL C64 when I remove all sanity checks. It does cheat a little, but if you can't notice it that doesn't matter, does it? NTSC C128 should do 30 Hz easily - I want one!

POSTED BY TNT AT 7:29 PM NO COMMENTS:

SUNDAY, AUGUST 12, 2007

Objects want to be free - but only be freed once

This weekend wasn't all bad in spite of having both my birthday and wedding anniversary (yes, my wife wanted to make sure i wouldn't forget the latter! ;) as I found the reason for the double-free bug - or at least one of the reasons.



I outlined the collision check loop(s) in an earlier entry. What's not shown in that code snippet is caching of outer loop object type, done to speed up the collision type decision. Even when the outer loop object gets removed in the collision the inner loop still continues afterwards. Check out the picture: the laser bolt colliding with two explosions is the object which gets removed twice. If the outer object type wasn't cached then all remaining collisions with it would be NOPs.

I have at least three ways to fix this:

1. don't cache the object type
2. exit the inner loop when outer loop object gets removed
3. check object type against -1 (free object) every time before freeing one

I will do #2 although it means duplicating some code, as I hate to waste cycles for checking for special cases unless it's absolutely necessary. I will rethink my decision if object removal still bugs.

POSTED BY TNT AT 9:52 PM NO COMMENTS:

WEDNESDAY, AUGUST 8, 2007

Hum Bug Betatesters

No one tells me about bugs... so I just have to play the game to find them myself! :)

- Droid library had two bugs: every droid class text was the player class, and droid entries went from 0 to 23 instead of 1 to 24.
- Droid count in console was wrong, it ignored some droids.
- Deck layout horizontal scroll was broken. You all **did** know you can scroll it now, didn't you?
- Maintenance deck had one droid too many on the left part, one too little in the right one. This one you really should have noticed!
- Remaining bugs:
  - Radar is borken.
  - Lift ignores short fire button press just after up/down move.
  - Game tries to free same object twice, this causes a freeze when my sanity check catches it.

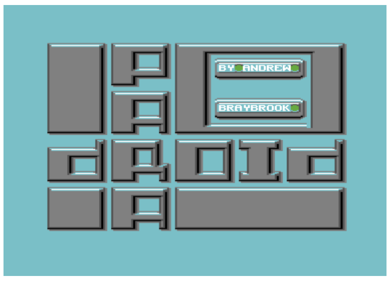
I bet there are more, but the last one is the most serious one. Thanks to trurl for first spotting it, now I can cause it too. The easiest way to do it is to enter ship 8 upper cargo when alert is red, then wait in the middle of bottom left room when berzerk droids circle around you shooting like there's no tomorrow. Well, without cheating there is no tomorrow for you...

Oh yeah, forget about the constant energizer animation part of the previous entry - I dropped it ages ago. I just hadn't played any of the official versions for a while, so when testing the Metal Edition it struck me as a difference to Competition Edition.

POSTED BY TNT AT 8:36 PM NO COMMENTS:

SATURDAY, AUGUST 4, 2007

Morphing into multicolor mode



I made a quick disassembly of Paradroid Metal Edition to see how much I need to change the code to make it possible to use either hires or multicolor graphics. The answer: not much. Color tables are different because only the lowest eight colors are available for character color and because of extra entries for multicolor registers. To use ME graphics I only have to reserve some extra space for the bigger tables and new title screen and add minimal amount of code. As I can do the patching inside the init routine which gets overwritten when game starts this will cost less than 80 bytes. The only thing that concerns me is the time taken by constant energizer animation, that's almost 4 lines more than changing the deck map. Does anybody notice if I drop it? ;)

The disassembly also showed that while Competition Edition was a quick hack to get something out for Xmas (in a double pack with Uridium Plus) Metal Edition got some more attention. Andrew Braybrook removed code which was disabled in the Competition Edition (raster checks which made sure that the original didn't run faster than 16 Hz), and enabled 2 MHz mode in the upper/lower border for C128 users. He also fixed the decimal mode bug which broke SFX randomly and adjusted background sounds for the faster frame rate, among other small things. Some of these changes will end up in Paradroid Redux, no doubt about that.

Too bad that I dislike Morpheus-like graphics in Paradroid myself. Anyway, those people who think that Metal Edition has the "correct" graphics can soon enjoy Paradroid Redux as much as fans of the original.

POSTED BY TNT AT 10:41 PM NO COMMENTS:

FRIDAY, AUGUST 3, 2007

Weekend fun

After recovering from the last weekend I finished the latest changes. I hope I didn't break anything in the process... Check it out yourself.

- **Lots** of flicker removed when changing screens.
- Some internal changes which makes it run faster, although it **still** drops to every 3rd frame under extreme situations.
- Game now knows about droid visibility changes which allows some more AI.

This version has most of the sanity checks removed, so if older routines get confuzed they will trash random memory. Only the droid allocator has code to hang the game "nicely" if it fails unpredictably.

Currently visibility change just flashes droid color. As you can see - or rather as you **can't** see - flash is only visible when going around corners and through doors. This means that droids approaching from distance have unfair advantage of spotting you before you spot them. This needs to be changed so that at least part of droid has to be visible on screen before it reacts to your presence.

Radar equipped droids won't get surprised as easily as other droids, and as alert status rises every droid with brains will react faster. I'm thinking of using different delay values for direction change, depending on droid's weight and drive. Surely 28 kg anti-grav 821 can stop and turn faster than 227 kg bipedal 751!

POSTED BY TNT AT 9:21 PM NO COMMENTS:

TUESDAY, JULY 24, 2007

Cleaning up

Now that the worst bugs seemed to be fixed, I took some time to fix most of the flickering when the game changed screens. I did that once already, but delaying sprite writes made the flickering come back. Now everything seems to be fine except going to game mode, where sprites are delayed that one frame. I'm not sure if I bother to fix that or not - it's hard to notice the delay without slowing the game down in emulator.

I've spent enough time doing boring tweaking, next day or two will be spent playing the game instead. There are loads of sanity checks to catch the bugs which should be gone now, so playing will be purely for quality control ;)

POSTED BY TNT AT 9:59 PM 2 COMMENTS:

The mother of all bugs killed

It seems that I finally nailed it! Whenever the collision check had to swap sprite indexes (to avoid having separate routines for droid-laser and laser-droid collision, for example) it also swapped counters for outer/inner loops.

```
    lda $d015       ; active sprites
    and #$7e        ; mask out player and player_fire
    sta mask

    ldy #7
.outer_loop
    asl mask
    bcs .has_high_sprite
.back_o
    dey
    bne .outer_loop
    rts

.has_high_sprite
    beq .back_o     ; no sprite to collide with

    sty high_sprite ; remember sprite count

    lda mask
.inner_loop
    dey
    asl
    bcc .inner_loop

    pha         ; remember all remaining sprites
    sty low_sprite

;   do stuff

    ldy low_sprite
    pla
    bne .inner_loop

    ldy high_sprite
    jmp .back_o
```

That one has no chance of working if low_sprite and high_sprite get swapped. Inner loop starts accessing too high sprites, and outer loops messes with lower sprites than intended, eventually accessing sprite 0 (player) and negative numbered sprites (completely random data).

This shows a situation where HLL compilers beat humans when generating code. Human with limited memory can't remember which routines read/write which variables, leading to local variables having way too long life span. Compilers know exactly which locations are needed and when, and that allows them to reuse local variable area much more efficiently.

BTW, that's my only admission to the mantra compiler writers repeat: "modern compilers write as good code as any human". I have seen some pretty impressive compilers for embedded systems, but I have seen them beaten, too ;)

POSTED BY TNT AT 12:01 AM NO COMMENTS:

MONDAY, JULY 23, 2007

Another Droid in the Wall - not any more!

As expected the game has some trouble running after the changes in droid allocation, but that did make me disable all collisions (which are the only remaining thing bugging because of allocation change) and enabling them one at a time. This pinpointed droid reversal bug into droid-droid collision.

Simplified droid movement logic is as follows

- Move droid unless it's paused.
- If pause count is not NULL, decrease it.
- If pause count is still not NULL, exit movement phase.
- If droid is on waypoint, pause it or give it new x/y speed.

The collision logic for droid-droid logic may also reverse droid direction. If that happens during the game frame when droid was given new direction from waypoint, the droid ends up in the wall unless it was lucky and ends onto another waypoint by sheer luck. To avoid that game checks if droid is on a waypoint and pauses it instead of reversing it in that case.

Waypoint check needs droid character position to compare against waypoint positions. Guess what? Droid char position isn't properly set at that point! Doh!

There are couple of ways to fix this

1. Calculate char pos for the droid to be reversed.
2. Set flag for each droid, telling if it was on a waypoint in a movement phase. That avoids the check altogether.
3. Add two more attribute bytes to each droid, and store the char pos there.

I don't like option 1, as that may require multiple calculations per droid for one game frame. Option 2 is good, as it avoids waypoint check altogether. This saves one to three raster lines if droid needs reversing. Option 3 is good too, as it is guaranteed to save at least one 44-cycle subroutine call per droid. Calculating char pos for explosions is extra work, but explosions are much cheaper than any other object types so that evens it out.

Forgetting to set droid char pos wasn't the only stupid error... My collision check determines colliding object types by calculating index into offset table, then doing calculated jump like this:

```
    lda     jumpoffset,x
    sta     jump-1
    bpl     *
jump


jumpoffset
    dc.b    droid_droid-jump,droid_laser-jump,...
```

It would have worked much better if assembler told me that jump offsets got big enough to get negative... This means that the current beta version skips some collision checks.

With those bugs fixed I just need to find out what's wrong with collisions involving droid fire. At least one of them messes up my droid_to_sprite and sprite_to_droid links.

POSTED BY TNT AT 11:04 AM NO COMMENTS:

SATURDAY, JULY 21, 2007

It's working - time to break it!

After couple of changes the game now seems to run solid 25 Hz on PAL C64 (and should do 30 Hz on NTSC C128 but I have no means to verify that), but to make sure it stays that way I'm finally replacing the old droid table with a proper droid allocator à la Mike. That avoids copying data every time an object is removed.

Direct result of that is that I can freely add more data for each droid as I don't have to worry about removal time. I have some ideas how to use that to speed up the game and add new features, like droid behavior when its visibilty changes. Now droids actions between waypoints are limited to whether to fire or not, flee/pursuit decision is only done at waypoints. I want to make droids more intelligent, even if it's nothing more than pause and then reverse their direction when visibility changes. That way 123 entering a room when player controls 999 will get second thoughts about going where it was intending to, and retreats instead.

I hope that the current beta works enough to keep you happy, it may take a while before the game works again...

POSTED BY TNT AT 6:55 PM NO COMMENTS:

FRIDAY, JULY 20, 2007

Speeding causes fatal crash in space!

"Did you really think you could run 1.3 MHz and not crash, sir?" the C128 cop asked me after the game went awry on my FrankenDiesel.

It's true: the game was **too fast** on C128! What happened was that at the beginning of a game frame the main loop told IRQ *"Grab sprite register data next display frame and stuff them into VIC-II when I'm busy minding other business, please"* and went on to do its other business. The screen redraw takes almost a complete display frame on PAL, more than one frame on NTSC, so everything was ok. But with C128 running 2 MHz in the lower/upper border screen redraw finishes a lot before the IRQ grabs the sprite data. In fact it finishes so early that the main loop can sometimes run the complete game loop before IRQ gets to do its task. An there the disaster strike: the main loop told the interrupt *"do it next frame"* **before IRQ had done the previous task** and so sprite registers never got updated.

I fixed this with a working frame limiter, forcing the game use at least two display frames for each game frame. There was a limiter before, but that broke down when I added VIC-II buffering. I bet it knew that it could cause havoc that way...

But wait - C128 problem isn't fixed yet! Because main loop still runs droids before IRQ grabs the sprite data, sprites may get the position meant for the next game fram, and they start to wobble around the screen again. And that's what I was trying to avoid in the first place when I added the buffered writes.

Solution was surpisingly easy: screen drawing finishes somewhere around raster line 0 on PAL C128, later on all other computers. So I can do the sprite register writes in the bottom-of-the-display-IRQ which is guaranteed to come before any droids get run. This has the added bonus that now the copy loop runs in 2 MHz, speeding up the game ;)

Another bonus is that now I can allow the game run every frame if I really want. If I ever write C128-specific version I can save ~4000 cycles in the screen draw alone by unrolling the loop - that takes "only" 8 KB or so. In the end C128 version might run at 50 Hz all the time - Paradroid Redux Competition Edition, anyone?

Update

Bugfix at the usual place. Title screen crashed randomly on C128 too, that should be fixed now.

POSTED BY TNT AT 11:56 AM NO COMMENTS:

MONDAY, JULY 16, 2007

Double the fun!

Rather quick new release, fixing some things:

- Subgame didn't clear it's local variables before retrying after deadlock.
- Background stars flickered on NTSC, now screen/star draw is done in two parts so stars get updated in time.
- "Floating" player fire hopefully fixed by caching joystick position through the routine.
- Sound routine tweaked for sharper sounds, unfortunately this has changed couple of sound effects.

If the game hangs it has most likely tripped on one of my sanity tests. Freeze the game and tell me the program counter so I can check where that happened.

POSTED BY TNT AT 11:16 PM 4 COMMENTS:

? Deadlock error, please debug

I found the bug which broke the transfer game. Some time ago I got rid of routine which finds starting locations of tokenized strings and stored them into a temporary table for later use, and replaced it with realtime search. There was no speed difference anyway because of raster sync and that saved some memory. Not only in the printing routines, but I could remove now unnecessary temporary buffer clearing from multiple places.

Unfortunately one of those places was in the subgame, in the retry loop to be excact. Ok, it worked perfectly unless you got deadlock and there were auto pulsers active. In that case temporary buffer still had auto pulser data active and copied color from now gone pulser position to the central bar. Usually there was wire, resulting in black color. Fix was simple: put that buffer clear back.

I've been tweaking the sfx routine further. Now it takes about 8 lines at maximum and sound starts are clearer. That however intoduced nasty snapping sound to repeating sounds, especially transfer fx. I think I can prevent that by checking if the new sound is the same as the previous one, and skipping gate off in that case.

POSTED BY TNT AT 6:35 PM NO COMMENTS:

SUNDAY, JULY 15, 2007

Sound of optimization

I went through the sound effect routine and optimized it a bit. As a result it's slightly faster, but what's more important is that now I can move all sfx zero page variables to "normal" memory and that will only cost one or two raster lines. That means that I have over 30 ZP locations which I can use for other purposes. Too bad that VICE doesn't have memory access profiler, as with one it would be easy to find out which variables/tables are accessed the most. Counting label occurences in source files gives some hints, but that doesn't count loop iterations etc. Time to do some VICE hacking I guess. Unless I find something more interesting to do, of course. Running out of interesting things made Mike add weapons systems to XeO3, I'm facing the same problem here!

Well, I have the droid allocator to rewrite too. The problem with the current method is that there is **no** allocator at all, so every time droid/laser/explosion is removed, every object behind it in the list needs to get moved down in memory. Changing this will make object removal a breeze and solves some other problems at the same time, but changing it also means that everything is broken for at least a couple of days because of things I can't even imagine yet. So I've stayed away from it. Well, I have to face it one day anyway...

POSTED BY TNT AT 8:52 PM NO COMMENTS:

SATURDAY, JULY 14, 2007

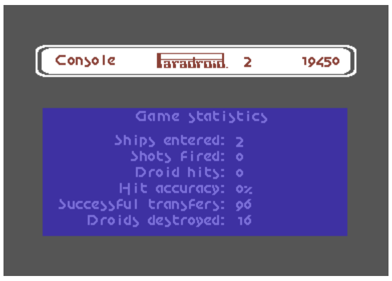
It's playtime!

Grab the new beta and have some fun.

Changes since the last release:

- Decks have separate start position, you no longer start new ship on one of the waypoints.
- Several waypoints near lifts have been excluded from available droid positions, making it safer to enter some decks.
- Deck sections are completely separate from each other.
    - Droids stay in the correct deck section between visits and if teleported away.
    - Power off condition is now done for each section separately.
    - Bonus is still awarded only when the whole deck is cleared.
- Some twinky stars added to background. Needs more work, especially for NTSC.
- Droids aren't completely predictable under red alert any more.
- Player droid damage calculation modified slighly, now higher class droids can take couple more hits under player influence.
- Fixed character animation, ended up rewriting it completely.
- Improved Game Over TV static.
- F7/F8 selects starting ship, assumed you've cleared one or more ship.

There may still be a problem with the subgame, but I completed the first ship without shooting a single shot (pacifistic takeover?) without seeing the bug so at least it isn't too frequent. It might have gone away when I added some more temporary buffer clearing.



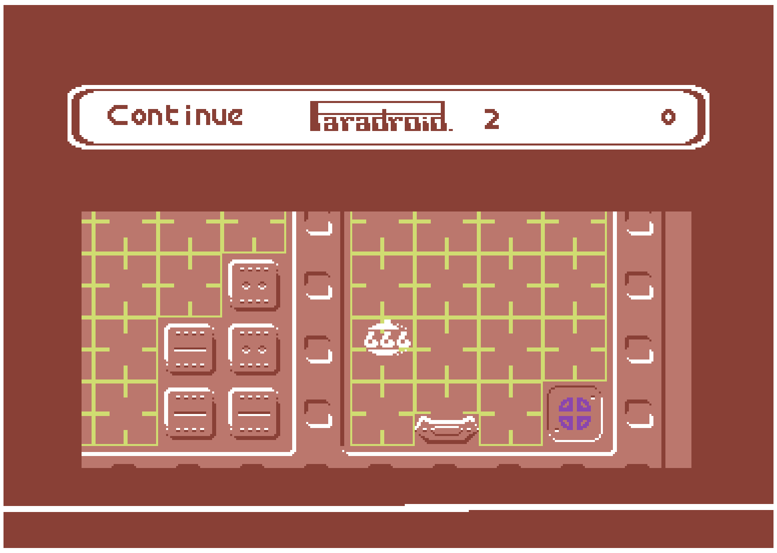
```
Console    Paradroid  2     19450

        Game statistics
        Ships entered: 2
          Shots fired: 0
           Droid hits: 0
         Hit accuracy: 0%
 Successful transfers: 96
     Droids destroyed: 16
```

POSTED BY TNT AT 4:38 PM 4 COMMENTS:

FRIDAY, JULY 13, 2007

It works again!



After some cursing, hair tearing and generic despair it's working again. Well, it awards deck bonus of 500 points when you clear deck section even when there are droids left in the other section, but that's a minor problem. The most important thing in that is that it now sees section as cleared and knows to turn off the lights,regardless of remaining droids. That means that droids stay put in correct half without teleporting between sections and droids in other sections aren't active at all.

Droids also seems to obey my new "don't start on this waypoint" system, which was evident when I tried entering bridge. I had excluded one waypoint too many, so there wasn't enought starting points for all droids. The game hung when trying to find free waypoint for the last droid. I like nothing more than clear signal when something is wrong :)

New waypoint checking system is a lot faster than the original one. The original check used max. 11 lines every time droid was centered on tile, no matter if there was a waypoint or not. And because of a bug it did it twice per tile! It scanned through Y-sorted list until it found same/higher Y coordinate. As there are only 14 valid Y positions for waypoints it meant that multiple X coordinates needed checking too.

Now every waypoint is marked with magic char, so droids only check for waypoint when they are over one. First check is the the usual droid_centered, and only if that's true game proceeds to find which waypoint data is needed. That test uses binary search to find the correct waypoint among 31 possible ones, so there are max. 5 comparisons.

To cut needed time down even more game calculates simple hash from x & y coordinates and uses that as primary search criteria. Only if that matches (and there are only eight hash collisions total so it doesn't happen often) it needs to test for another byte. And remember - this check will match in the end, it is never done unnecessarily.

Max. time taken is slightly over three raster lines, as seen in the attached picture.

Next beta will be out this weekend.

POSTED BY TNT AT 11:55 PM NO COMMENTS:

THURSDAY, JULY 12, 2007

All hail simplicity

After I wrote the code to create all droids on ship I had second thoughts about it. The code was horrible mess, parsing deck data to get correct number of certain level droids into correct sections. Nested loops, lots of comparisons, checking every section to get droid distribution right. In the end I made one 256 byte table to place every droid where it belongs. 256 bytes is quite a lot of memory compared to 30 bytes which the droids required when associated with decks and sections. However, the table ended up being only 48 bytes when packed, and the routine which goes through it (once!) was about 80 bytes shorter than my original code. In fact it was shorter than the one from original Paradroid, and that one didn't know anything about deck sections.

Now the ship contained all the necessary inhabitants, so it was time to check out the code which activates correct droids when player enters a deck. That was bound to be easy. However, when I depacked section waypoints I didn't store their Y coordinates at all. This confused droid placement, and it had every reason to do so. That meant that I had to put Y coordinates into a temporary buffer where InitDeckDroids() could find them. Yeah right, except that there was a routine between the two subroutines which cleared the coordinate table...

POSTED BY TNT AT 10:58 PM NO COMMENTS:

WEDNESDAY, JULY 11, 2007

NewsFlash! Viral infection grinds PR development to a halt!

Not my computer, luckily. Most of the last two days has been used to salvage data from one of the worst infected computers I've seen. Booooooooring! My recommendation: buy removable HD. Copy data there. Format C: Reinstall.

I've managed to teach the game quite a bit more about ship layout, droid distribution and other small things. Now it knows which deck part lifts connect and **really** should be able keep droids in their respective deck parts. I bet tomorrow will be spent on debugging every possible problem caused by game not generating droids, droids jumping around whole ship, waypoints being completely garbled up and whatever the game can think of.

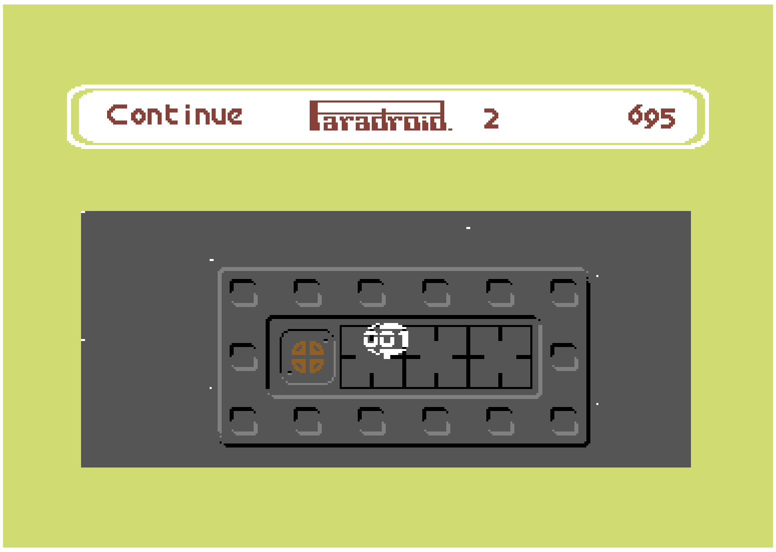
Tidbit of the day: left side of observation has two waypoints, but game never generates droids there.

POSTED BY TNT AT 11:08 PM NO COMMENTS:

TUESDAY, JULY 10, 2007

My God, it's full of stars!

I'm not sure if I like stationary background stars (this time I nicked the idea from Uridium) as wall chars have couple of empty pixels at the edges. This means that stars will disappear before they reach the ship. This could be explained with diffraction in the micro-atmosphere around the ship, of course...



Adding stars would be quite cheap - I only need to check for underlying char being "outside_of_deck" and replace that char with animated star char. Actually I check the next one too, to be able to have two pixels wide stars. Animating the star chars is even easier:

```
.stx
    ldx #0
    lda #0
    sta Font+chr_STAR*8,x
    sta Font+chr_STAR*8+8,x

    lda vScroll
    eor #7
    tax
    stx .stx+1
    ldy hScroll
    lda bits,y
    lsr
    ora bits,y
    sta Font+chr_STAR*8,x
    lda #0
    ror
    sta Font+chr_STAR*8+8,x
```

First it clears previous star pixels, then calculates new Y position, stores it for the next round and plots two bits into chars.

Don't mind the regular position of stars, that's just test data. Stars are exactly 64 chars apart from each other. If this gets used then star positions get randomized every time you enter a deck.

POSTED BY TNT AT 12:07 AM 4 COMMENTS:

SUNDAY, JULY 8, 2007

Bit by bit

After I had to come up with a name for the fastloader I decided to post this bog-standard routine I use in Paradroid Redux to depack waypoint and droid distribution data. The only reason it's worth of posting is that it doesn't use any other register than accumulator.

```
;   in: ax=ptr

InitGetBits
    stx .get+1
    sta .get+2
    lda #$80        ; start with end bit
    sta cbits
    rts


GetBits
.1  asl cbits       ;  5
    bne .3          ;  8

    pha             ;( 3)
.get
    lda $1234       ;( 7)
    inc .get+1      ;(13)
    bne .2          ;(16) assume not taken
    inc .get+2
.2  rol             ;(18) rotate end bit in
    sta cbits       ;(21)
    pla             ;(25)
```

```
.3  rol             ; 10
    bcc .1          ; 13  assume taken
    rts
```

13 cycles per bit unless new byte is needed, 25 cycles extra every 8th bit. This adds up to 16.125 per bit on average, not accounting for jsr/rts overhead.

Calling it requires loading A with a bit mask telling it when to stop. Usually this is single bit to return only input data, for example %00010000 to read four bits. However, as I need to ORA the four-bit Y coordinate with #$20, I can do it for free by doing "LDA #%00010010; JSR GetBits". The loop ends when the highest bit is rotated into the carry, so A is "0010yyyy" at that point. Even better, after I've stored the ORed value I need to multiply A by four and then add two, and this time I can use the fact that carry is always set on exit so it's just "ROL; ASL". Try to do that with a high-level language :)

Edit: oh bugger, IE6 collapses multiple line feeds into one even inside PRE tag. I hope IE7 is more intelligent.

POSTED BY TNT AT 11:55 PM 1 COMMENT:

Loading, please wait...

I cleaned up XeO3 loader (yes, it's for Commodore 264 family) slightly and added couple of comments. You can find the loader + source code here, I hope it gets added to Plus/4 World soon so people can actually find it.

The loader works with 1541/157x and 1581. Every bit is clocked separately, so it's safe to use interrupts while loading. It beats many 2-bit loaders, and allows keeping the screen on.

Resident loader part is less than 256 bytes. Save works too, but maximum size is 252 bytes and file to overwrite must be on the disk already. It's only meant for high scores anyway ;)

Porting back to C64 is easy enough, just change couple of variable definitions and remove unnecessary code.

Update:

This is not a generic turbo you would use to load programs with. It was meant for game/demo use and so it has some limitations.

- You can't **LOAD "$",8**.
- Drive code is disabled (drive is reset) as soon as ATN line is active. This means that it's gone as soon as you access any device on IEC bus with kernal routines.
- Drive code is installed only once, so once it's gone, it's really gone. That's the dark secret behind small resident code.
- If you try to LOAD/SAVE without restoring kernal vectors once drive code is gone, computer will hang waiting for drive response.

Optimal interleave is 16 sectors, unless your interrupt takes lots of time. In that case try interleave 17 or 18 to see if that speeds up loading. Normal interleave used by 1541 is 10, which means that the loader must wait additional 10+ sectors to pass by the r/w head before reading next sector from disk surface.

POSTED BY TNT AT 8:52 PM 3 COMMENTS:

FRIDAY, JULY 6, 2007

629 - number of the crap

No, not really. 883 still holds that position, but 629 is a close runner-up. 615 is bit more usable after damage adjustment, but I'm still not going to change my playing style.

Lift exit bug seems to have gone away when I reverted back to the original code. I hope it stays there, too.

POSTED BY TNT AT 11:16 PM NO COMMENTS:

Aye, captain! Turbolift is up and running

One deck map is 16 KB of data, so it takes a while to build one from 64*16 byte layout. That's why lift movement between decks is as slow as it is - game builds the deck in advance to avoid delay when you exit the lift.

I modified the lift routine to check if it was allowed to move up/down from the new deck and to pass that info to the deck building subroutine. Deck builder checks halfway through the data if joystick is pushed to allowed direction, and if it is then the routine exits without finishing the deck. Lift routine then sees the return value and moves to next deck if needed. When joystick is finally released or there are no more decks above/below current one, the deck gets fully built.

The end result: lift appears to be twice as fast as before, without affecting the exit time much.

While testing it on real C64, I noticed that a new bug has crept in - sometimes the game pushes player droid inside the wall when it exits the lift. I haven't been able to repeat the bug inside VICE, which makes it "fun" to find. I suspect my new and optimized "keep old position on tile when exit" code. Maybe I will just center the droid on lift tile, as I bet there's "stay away from walls" sign in the lift anyway...

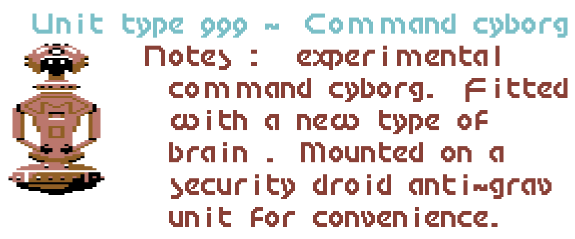
I've also modified player damage calculation slightly. Earlier player droid class didn't affect the amount of damage it received from enemy fire, now higher class droids take a little less damage, especially from single laser. The difference isn't big, but now it's a bit more tempting to use higher class (and faster decaying!) droids instead of 476.

POSTED BY TNT AT 8:35 PM NO COMMENTS:

THURSDAY, JULY 5, 2007

?No space in brain error

You all have noticed how 999 description text has extra space after the word "brain", haven't you? Twice, even. That's because the same string token is used for the first page of droid data, and space is needed to


Unit type 999 - Command cyborg
Notes : experimental command cyborg. Fitted with a new type of brain. Mounted on a security droid anti-grav unit for convenience.

align the text. Three other strings are padded too, but they don't affect any other text. Instead of adding one more token costing five bytes I removed all trailing spaces and added code to do the aligning. Couple of changes later I had won one byte and nice warm feeling, knowing that now the text is perfect.

Speaking of space saving - I checked how much free memory I have, and I have 400+ more bytes than two weeks ago. If I continue like this the game doesn't use any memory at all after couple of years...

... no, not really. New TV static routine suggested by Mike is longer than the original. Oh, horror! However, it looks **so** much better than the original that I can accept the loss of six bytes.

POSTED BY TNT AT 10:27 PM 2 COMMENTS:

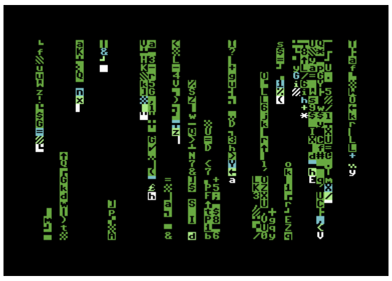
I took the blue pill

While the Matrix effect with Paradroid charset doesn't look all that bad (in fact I like it more with 1x2 charset) it looks really out of place when it's used inside the game. Next one to try is to get "true" random static. That may require more than four different "random" chars, which means some extra data to be moved around. I already re-use console icons for waypoint markers, swapping them in and out as needed.

POSTED BY TNT AT 7:00 PM NO COMMENTS:

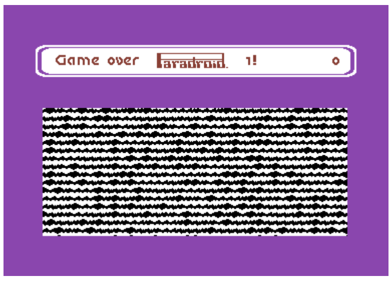
WEDNESDAY, JULY 4, 2007

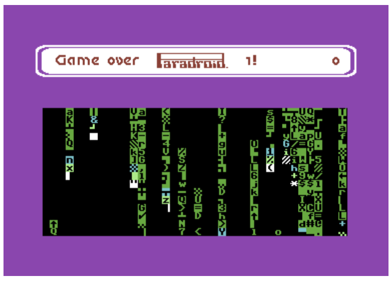
Too kitsch, or not too kitsch: that is the question.



Nothing much done today, except one completely unrelated routine inspired by some random web site. Funny thing, I have never thought about writing Matrix character drop effect for C64 before.

Now I'm wondering how it would look with Paradroid font. Core routine is about 200 bytes, which is longer than the original TV static effect. However, I've saved quite a lot of space reserved for data, so I could take some of it back.





I guess there's only one way to find out how it really looks. Code has to be changed slightly to cope with 1x2 chars, but other than that it should be simple case of inserting the new code (famous last words?)

POSTED BY TNT AT 8:57 PM 5 COMMENTS:

TUESDAY, JULY 3, 2007

Let There Be Light

On a sunny summer day in Finland it finally happened - I gave in and started a blog about my Paradroid hack. Unlike PC remakes this one is made for the Commodore 64. (Enough links already? Can I stop it now?)

The disassembly was completed over a year ago, and since then there have been many changes. Main goal was to give it a nice speed boost, but I couldn't resist changing some other things as well. You can read more about it here, but let's recap the list:

- Support for second fire button.
  No more accidental transfers.
- Number of droids remaining on deck/ship shown in console
  Nicked from Paradroid'90.
- Statistics for previous game can be viewed during intro sequence.
  Nicked from the same source...
- Start ship can be selected from any previously visited ships.
  One more from Paradroid'90. This info is saved to disk so you can carry on from your last game.
- Scoring changes: alert scoring is altered to match the original instructions, ship clear now has both accuracy and alert bonus (p90 is good for inspiration!). I also replaced rather stupid "lowest scre of the day" with all time high score, which is saved to disk.
- Droid AI has changed quite a lot: droids flee/pursuit depending on their level, energy and ship alert status. Radar equipped droids do that even if player is not visible.
- Damage calculation has changes slightly as well. Droids getting hit by friendly fire sustain same damage than from player, and explosion does sligthly less damage so it's easier to survive one. Explosions don't restart any more when hit by laser/disruptor, which makes them less deadly as well.
- Original game killed droids when there were no free sprites, I attempt to teleport them to some other part of deck.
- Droids bumping into each other behave randomly.

You all knew that all already, so let's end this entry with something useful to know. When writing minimal saving routine, be sure to clear memory location $02a1 beforehand. Otherwise you may get endless loop when kernal waits RS232 routines to release IEC bus. Here is the minimal version of system restore before load/save:

```
RestoreSystem
        pha

        ldx     #$80        ; store game vars
.1      lda     0,x
        sta     $bf00,x
        inx
        bne     .1

        stx     $02a1       ; avoid IEC_idle hang
        stx     $9d         ; no messages

        inc     $01         ; kernal in
                            ; IRQ will now use $0314
        pla
        rts
```

POSTED BY TNT AT 5:06 PM 8 COMMENTS:

Newer Posts Home

Subscribe to: Posts (Atom)