A game in six versions:

Fun Beyond GS

Olivier Goguel

Well, it's not these damn consoles that are going to stop us from we continue to have fun with our GS! Since the production of games by “Big Companies” is a bit down at the moment, the FTA and ToolBox Mag are taking take over. Having seen the idea pass furtively on a wheelbarrow (it seems that the initial idea comes from a Unix game by J.Geertson called “xcolumns”), we We decided to implement it on a machine for adults.

The principle of the game is a variation of Tetris: instead of piling up blocks so as to fill the holes, we take figures made up of three square blocks of color. These blocks, once fallen, disappear as soon as three of them of the same color _ are aligned, according to the rules of tic-tac-toe (horizontal, vertical, diagonal alignment in both ways). As a result, the blocks above fall into the holes thus released, by “gravity”: eventually, they will then disappear in turn, by chain reaction.

We cannot, like in Tetris, rotate the figure, but we can invert the order of the blocks (R key). The N key displays the next part. A . Built-in “Help” will tell you everything you need to know to play Fun Beyond GS.

Originality, Fun Beyond GS is offered in English and French versions, and under three forms each time: Prodos 8 application, GS/OS application, CDA. You choose- Take your version to assembly, which is done directly in one go. Evi- Well, it's with Merlin 16 that we do all this! You will find the complete source on the floppy disk, catalog /BEYOND), subcatalog /BEYOND.SOURCE.

GS/OS, Shut up! An extract from the source of

Fun Beyond GS, by Olivier Goguel

[Editor's note: We do not have the space in the magazine to print Olivier's entire source. YOU you will therefore find it in the /BEYOND catalog of the ToolBox Mag 4 floppy disk. add to Beyond GS the acceptance of lower case letters on the keyboard, for example. But we want to print a routine, marginal compared to the game, but particularly interesting: Olivier fought with GS/OS to make it do what he wanted, and not what Apple had planned. And guess who won?

It seems to us, however, that Olivier fought needlessly with P8 for the access path to its program: to our knowledge, the complete path of a SYS application which just launched is still in 00/0280 and the rest. Does anyone know if it's just a habit, or if this is guaranteed by the Political Bureau?]

Throughout this program, we will -as usual- manage everything by ourselves (display, error management) reurs, etc.). Unfortunately, we're going to be light- bothered by GS/OS, which has the possibility of displaying dialog windows (text or graphics) for us warn of the appearance of possible errors. Theoretical- ment, the SetSysPref call allows you to tell GS/OS not to not worry about error handling:

```
                JSL   SE100A8
                DA    $200F             GetSysPref
                ADRL  SysDefaultParms

                JSL   $E100A8
                DA    $200C             SetSysPref
                ADRL  SysParms
```

But in fact, in certain cases, GS/OS will -despite the pre- References requested - show some error windows: which would be fatal to our application as well as to system, whose cohabitation is impossible outside the moths defined by. Your humble servant !

We are then forced to patch GS/OS, without scruple- pule, since no approved appeal allows one to make exactly what we want. We are not going all the same not let yourself be dominated by any operating system conch….

So I "examined" part of GS/OS to go back up to the routine that interests me: this one is located ted by a vector (whose address, although not declared, seems immovable to me) in $01FC98.

If we follow what is pointed by this vector, we find a bit of routine that starts with:

```
*               LSR
*               BCS   AfficherFenctre
*               LDAL  $00B9F6           Test des préférences
*               BMI   AfficherFenetre   (Bit15=Affichage)
*               …
*               RTL
* Aff_Fenctre    …
```

We can clearly see the BCS which can force the display of a window despite the state of the Preferences flag...

We will therefore divert the vector at $01FC9S to a routine which resets the accumulator to 00, so that the BCS is never taken, and thus we will never have unwanted windows on the screen.

We will remove our patch when leaving, of course. In the meantime, it is our application which will handle errors, in the absence of any GS/OS message (in particular the "Insert disk XXX" message).

```
                SEP   $20
                LDAL  $E1C068
                PHA
                LDAL  $E1C08B
                LDAL  $E1C08B

                REP   $30

                LDAL  $01FC98     We verify that it is
                AND   #$00FF      indeed a vector.
                CMP   #$5C
                BNE   NoPatch

                LDAL  $01FC99     Backup of
                STA   SaveVect+1  the value of the vector
                LDAL  $01FC9A
                STA   SaveVect+2

                LDA   NewVect+1   and diversion
                STAL  $01FC99     towards our wheel
                LDA   SaveVect+2
                STAL  01FC9A
```

```
NoPatch           SEP   $20
                  PLA
                  STAL  $E1C068
                  REP   $30

                  BRA   Continue_Init
--------------------------

SysDefaultParms   DA    1           Backup
                  DA    0           Preferences
SysParms          DA    1
                  DA    0           GS/OS, shut up!
NewVect           JML   NewFC98     new value FC98
NewFC98           LDA   £0
SaveVect          JML $000000       old FC98 Value
```

(The rest on the floppy disk.)