go lucky

a board game for ATARI 8Bit computers

- how to create own levels -

introduction

Here you can learn how to create own levels for go lucky.

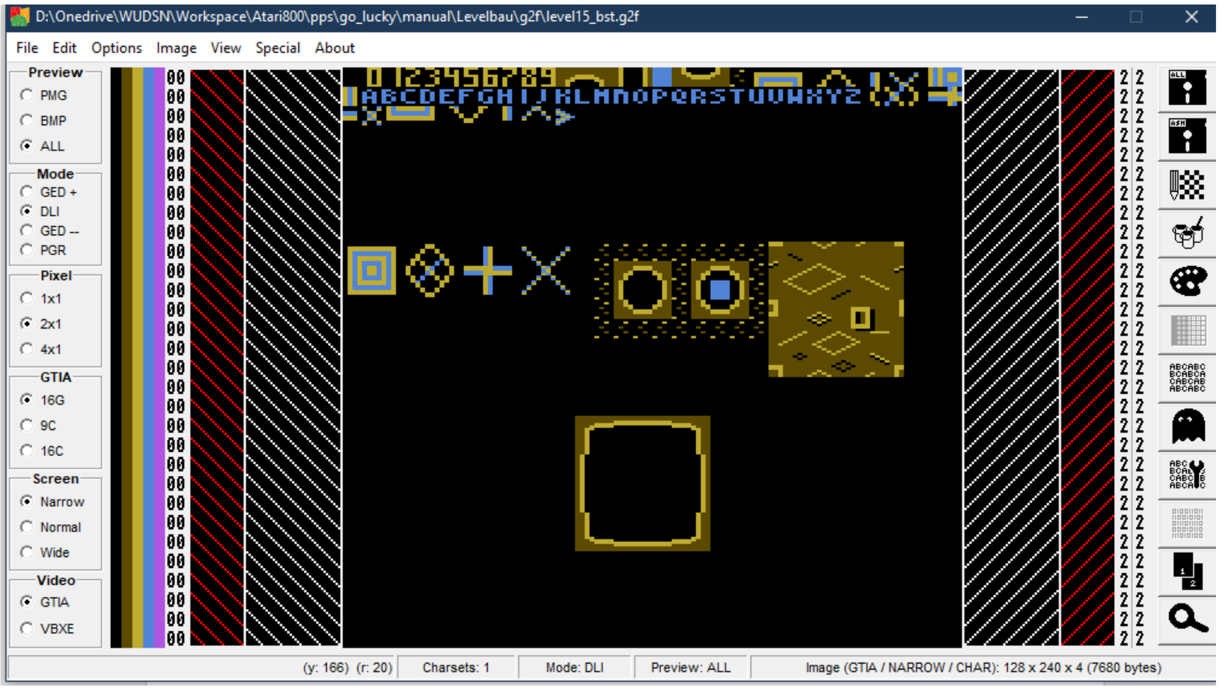
The game can handle up to 99 levels at one disc. As of the high demand of free memory I used uDOS as disc operating system. That gives us up to 256k discs. So, in theory, about 50 levels will be possible at one 256 kB disc at the end.

I used WUDSN (with MADS) and Graph2font to create the game.

table of contents

preparing a font

You can use any font creation tool to create a new font. To have some easy way to test, if the tiles will look nice, I use Graph2font here. Load the level15.g2f file to be able to edit the font.



D:\Onedrive\WUDSN\Workspace\Atari800\pps\go_lucky\manual\Levelbau\g2f\level15_bst.g2f

Preview: ○ PMG ○ BMP ● ALL
Mode: ○ GED + ● DLI ○ GED -- ○ PGR
Pixel: ○ 1x1 ● 2x1 ○ 4x1
GTIA: ● 16G ○ 9C ○ 16C
Screen: ● Narrow ○ Normal ○ Wide
Video: ● GTIA ○ VBXE

(y: 166) (r: 20) | Charsets: 1 | Mode: DLI | Preview: ALL | Image (GTIA / NARROW / CHAR): 128 x 240 x 4 (7680 bytes)

In the upper lines you can see the first parts of the font. Numbers and chars should not be moved inside the font. The full blue char is used as marker when you can change the investment money and at least in my levels for the warp tiles. The last char in these lines is the arrow used in the menus.

The area below is used to paint the players, tiles and background graphics.

I use a 7x7 raster to create the tiles, the players are 3x3.

Beside the players there is the old background tile around some of the tiles used as frame for menus etc. at right side there is the whole background tile and below that there is the frame of the other tiles.

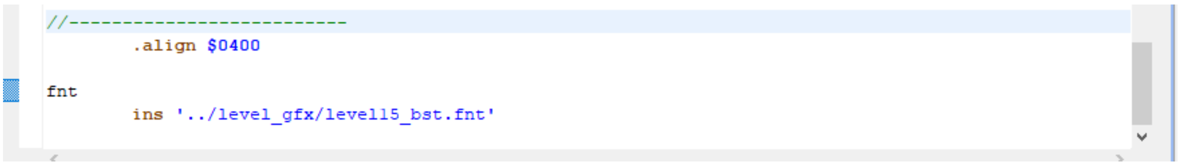
When you use Graph2font, pay attention that there is not more than one font used to show all, that you paint on the screen.

When you are ready, save the whole thing and then save the .fnt file by simply pressing save all button.

The created .fnt file needs to be copied to the asm tree/level_gfx folder (or correct linked to in the asm file).

edit the asm file

Load level15.asm from asm tree/data folder. At the very end of this file you can adjust the correct location of the new fnt file.



```
//-------------------------
        .align $0400

fnt
        ins '../level_gfx/level15_bst.fnt'
```

general structure of level file

definition of level size (in hex)

```
width       .he 09
height      .he 06
```

In theory we have up to 83 tiles max (4k border) - but this is not possible, so 9x9=81 will be real max of tiles. Always set these to correct level size, as the loading routine needs these values to load the level correct. Another thing is, that the game does not check the ‘resolution of your board’, so don’t use less that 06 x 04 resolution. For smaller boards you have to add some spare plots as seen in level 05.

money definition

```
startmoney              ;money at start
      .he d0 07 00 00   ;dez:=2000
aimmoney                ;money needed to win
      .he 40 9c 00 00   ;dez:=40000
```

The game uses 32bit, but if highest byte is set, we have negative values, so $7FFFFFFF is highest amount of possible aim - but that will not be shown as the game shows just decimal values with 6 digits, so $000f423F should be the absolute maximum aim. Both, start and aim money are stored from lowest to highest byte - $000007d0 := 2000. If you set start higher than aim, the first player will always win 😉

colors definition

```
l_colors
      .he d4 da 88 58 00      ;PAL Colors 0,1,2,3,bg
      .he e4 ea 98 68 00      ;NTSC Colors 0,1,2,3,bg
p_colors
      .he 76 1a c8 24 00      ;PAL Player Colors 0,1,2,3,4,placehold
      .he 86 2a d8 34 00      ;NTSC Player Colors 0,1,2,3,4,placehold
m_color .he 04                ;PAL Color for active Player marking
      .he 04                  ;NTSC Color for active Player marking
```

Colors of playfield (as defined in g2f) are l_colors and you can set the correct values for PAL and NTSC here.

p_colors handles player colors, m_color sets the color of the marker color in the status lines that marks active player. The marker to show you where to go is always the same color as color3 - it’s player 5 of pm graphics.

level definition

```
l_feld
```

what tiles will be shown

```
l_richtungen      ;binär wie Joystick ->
      ;1-links 2-rechts 3-hoch 4-runter, also bit 0&1 für re+li oder bit
4&2 für re&ru etc
```

where can you go

```
l_kategorien
```

what categories are the tiles

```
l_bereiche  ;Bereiche in denen Miete teurer werden könnte
```

areas the tiles belong to

```
l_ipreis_l  ;initial prices
l_ipreis_h
```

initial price of the tiles (low and high, 16 bits only)

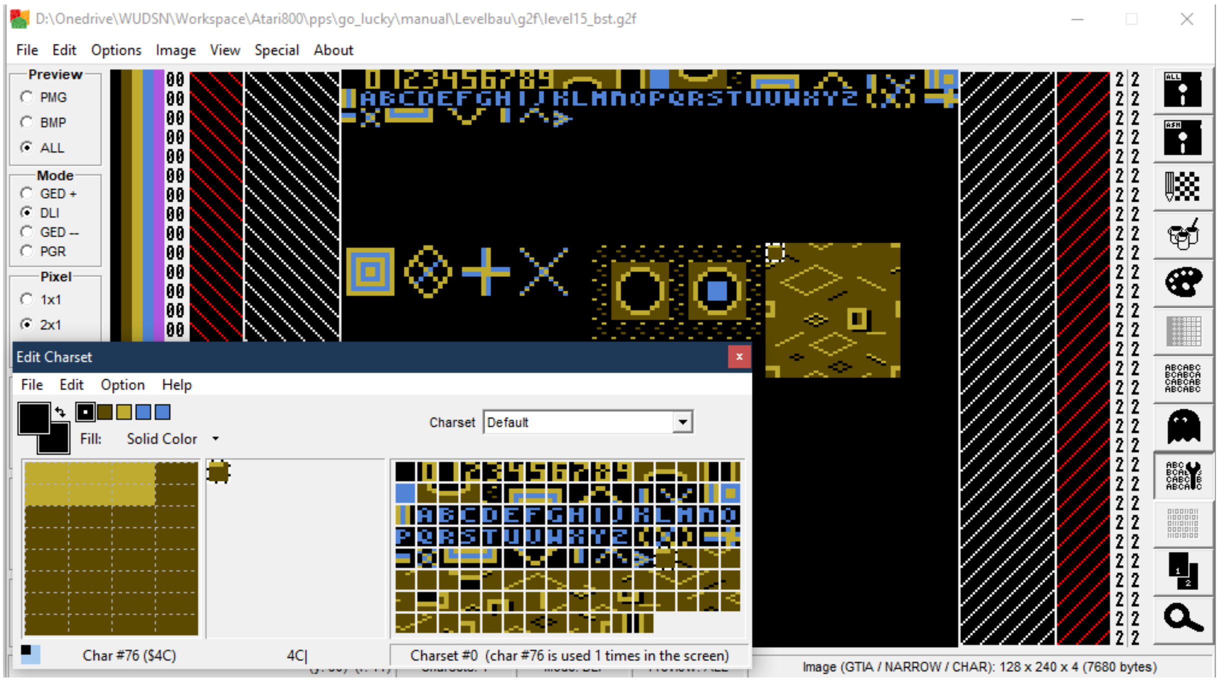
tiles definition

```
;------------ start of playfield tiles, each of 7x7
tile_bg     ;00
      .he 4c 4d 4e 4f 50 51 52
      .he 54 55 56 57 58 59 5a
      .he 5b 5c 5d 5e 5f 4e 60
      .he 62 51 63 59 64 65 66
      .he 51 67 68 69 6a 6b 6c
      .he 6d 6e 6f 70 71 72 51
      .he 73 74 75 76 77 78 79

tile_laden  ;01
      .he 0B 0C 53 53 53 0C 0D
      .he 7a 00 00 00 00 00 7b
      .he 0E 00 00 00 00 00 0F
      .he 0E 00 00 00 00 00 0F
      .he 0E 00 00 00 00 00 0F
      .he 7a 01 01 01 01 01 7b
      .he 11 12 61 61 61 12 13

tile_start  ;06
      .he 0B 0C 53 53 53 0C 0D
      .he 7a 00 10 10 10 10 7b
      .he 0E 10 00 00 00 00 0F
      .he 0E 00 10 10 10 00 0F
      .he 0E 00 00 00 00 10 0F
      .he 7a 10 10 10 10 00 7b
      .he 11 12 61 61 61 12 13

tile_luck   ;07
      .he 0B 0C 53 53 53 0C 0D
      .he 7a 00 90 90 90 00 7b
      .he 0E 00 90 00 00 00 0F
      .he 0E 00 90 90 00 00 0F
      .he 0E 00 90 00 00 00 0F
      .he 7a 00 90 00 00 00 7b
      .he 11 12 61 61 61 12 13

tile1       ;02
      .he 0B 0C 53 53 53 0C 0D
      .he 7a 00 00 10 00 00 7b
      .he 0E 00 10 90 10 00 0F
      .he 0E 10 90 90 90 10 0F
      .he 0E 00 10 90 10 00 0F
      .he 7a 00 00 10 00 00 7b
      .he 11 12 61 61 61 12 13

tile2       ;03
      .he 0B 0C 53 53 53 0C 0D
      .he 7a 90 00 00 00 90 7b
      .he 0E 00 10 00 10 00 0F
      .he 0E 00 00 90 00 00 0F
      .he 0E 00 10 00 10 00 0F
      .he 7a 90 00 00 00 90 7b
      .he 11 12 61 61 61 12 13
```

```
tile3       ;04
      .he 0B 0C 53 53 53 0C 0D
      .he 7a 10 00 00 00 10 7b
      .he 0E 90 10 00 10 90 0F
      .he 0E 90 90 10 90 90 0F
      .he 0E 90 10 00 10 90 0F
      .he 7a 10 00 00 00 10 7b
      .he 11 12 61 61 61 12 13

tile4       ;05
      .he 0B 0C 53 53 53 0C 0D
      .he 7a 10 90 10 90 10 7b
      .he 0E 90 00 00 00 90 0F
      .he 0E 10 00 00 00 10 0F
      .he 0E 90 00 00 00 90 0F
      .he 7a 10 90 10 90 10 7b
      .he 11 12 61 61 61 12 13

tile_warp   ;08
      .he 0B 0C 53 53 53 0C 0D
      .he 7a 00 90 90 90 00 7b
      .he 0E 90 00 00 00 90 0F
      .he 0E 90 00 90 00 90 0F
      .he 0E 90 00 00 00 90 0F
      .he 7a 00 90 90 90 00 7b
      .he 11 12 61 61 61 12 13

tile_warp2  ;09-11
      .he 0B 0C 53 53 53 0C 0D
      .he 7a 00 90 90 90 00 7b
      .he 0E 90 00 00 00 90 0F
      .he 0E 90 00 10 00 90 0F
      .he 0E 90 00 00 00 90 0F
      .he 7a 00 90 90 90 00 7b
      .he 11 12 61 61 61 12 13
;----------- end of playfield tiles

tile_cube   ;3x7
      .he 0b 0c 0d
      .he 0e 00 0f
      .he 0e 00 0f
      .he 0e 00 0f
      .he 0e 00 0f
      .he 0e 00 0f
      .he 11 12 13

tile_menu   ;8x11
      .he 0b 0c 0c 0c 0c 0c 0c 0d
      .he 0e
      dta d"  DICE"
      .he 0f 0e
      dta d"  VIEW"
      .he 0f 0e
      dta d"  SELL"
      .he 0f 0e
      dta d"MONEY"*
      .he 00 0f
      .he 0e 00 00 00 00 00 00 0f
      .he 0e
      dta d"PLOTS"*
      .he 00 0f
      .he 0e 00 00 00 00 00 00 0f
      .he 0e
      dta d"TOTAL "
      .he 0f
      .he 0e 00 00 00 00 00 00 0f
      .he 11 12 12 12 12 12 12 13

tile_bonus  ;8x9
      .he 0b 0c 0c 0c 0c 0c 0c 0d
      .he 0e
      dta d"BONUS"
      .he 00 0f
      .he 0e
      dta d"PLOTS"*
      .he 00 0f
      .he 0e 00 00 00 00 00 00 0f
      .he 0e
      dta d"ROUNDS"*
      .he 0f
      .he 0e 00 00 00 00 00 00 0f
      .he 0e
      dta d"TOTAL"
      .he 00 0f
      .he 0e 00 00 00 00 00 00 0f
      .he 11 12 12 12 12 12 12 13

tile_msg    ;8x4
      .he 0b 0c 0c 0c 0c 0c 0c 0d
      .he 0e 00 00 00 00 00 00 0f
      .he 0e 00 00 00 00 00 00 0f
      .he 11 12 12 12 12 12 12 13

tile_dialog ;8x6
      .he 0b 0c 0c 0c 0c 0c 0c 0d
      .he 0e
      dta d"BUY IT"
      .he 0f
      .he 0e 00 00 00 00 00 00 0f
      .he 0e 00 00
      dta d"YES"*
      .he 00 0f 0e 00 00
      dta d"NO"*
      .he 00 00 0f
      .he 11 12 12 12 12 12 12 13

tile_fortune      ;17x3 top,line2,linex,line before bottom,bottom of menu
      .he 0b 0c
      dta d"F",d"O"*,d"R",d"T"*,d"U",d"N"*,d"E"
      .he 0c
      dta "W"*,d"H",d"E"*,d"E",d"L"*
      .he 0c 0d                                ;top line
      .he 0e 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 0f     ;line2
      .he 0e 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 0f     ;linex
      .he 0e 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 0f     ;line
before bottom
      .he 11 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 13     ;bottom
line

arrow .he 4b

spieler1
      .he 15 16 17
```

```
        .he 1e 1f 20
        .he 42 43 44
spieler2
        .he 18 19 1a
        .he 3b 3c 3d
        .he 45 46 47
spieler3
        .he 00 1b 00
        .he 3e 3f 40
        .he 00 48 00
spieler4
        .he 1c 00 1d
        .he 00 41 00
        .he 49 00 4a
marker
:9      .he 10
```

This area sets the correct values of the font chars for the tiles we want to show. First part is to define the level tiles, started by tile 0, the background tile(tile_bg). Each tile is 7x7 and should be set to the correct char in the font. Graph2Font can help here with char edit window. You can select one char and see, what hex value it is.



D:\Onedrive\WUDSN\Workspace\Atari800\pps\go_lucky\manual\Levelbau\g2f\level15_bst.g2f

Preview: ○ PMG ○ BMP ● ALL

Mode: ○ GED + ● DLI ○ GED -- ○ PGR

Pixel: ○ 1x1 ● 2x1

Edit Charset

Fill: Solid Color

Charset: Default

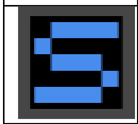
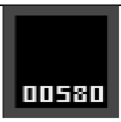
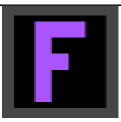
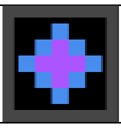
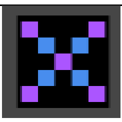
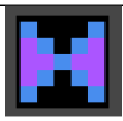
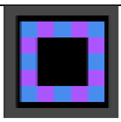
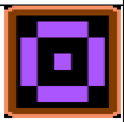
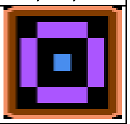
Char #76 ($4C)   4C| Charset #0 (char #76 is used 1 times in the screen)

Image (GTIA / NARROW / CHAR): 128 x 240 x 4 (7680 bytes)

tiles overview

tile_bg is the background tile and is $00 in l_feld.

first line title in code, second line gives possible value for l_feld

tile_start	tile_laden	tile_luck	tile_1	tile_2	tile_3	tile_4	tile_warp	tile_warp2
06	01	07	02	03	04	05	08	09,0a,0b
								
start	plot	fortune wheel	special item 1	special item 2	special item 3	special item 4	warp tile 1	warp tile 2
- start	- buy - rent	- spin it	got tile	got tile	got tile	got tile	warp, if got	warp by run past

- get bonus - win	- sell						direct on tile	

tile_warp and tile_warp2 always must be there in pairs.

tile_cube is the shape of the dicer. arrow is the char used to show the arrow in the menus. Spieler1 till Spieler4 are the shapes of the players. marker is used to mark a tile during sell. The char that marks the number to change during investment is always $10.

font include

```
      .align $0400

fnt
      ins '../level_gfx/level15_bst.fnt'
```

level definition

For original level15 we have following l_feld:

```
l_feld        .he 01 01 00 00 09 05 01 01 0a
              .he 02 00 01 00 00 00 00 00 00
              .he 01 00 00 03 00 00 00 01 01
              .he 01 00 00 01 01 01 04 00 01
              .he 01 00 00 07 00 00 01 00 09
              .he 06 01 01 00 00 00 01 01 0a
```

The values represent the tiles, we want to show.

Now we have to set, where we can go:

```
l_richtungen        ;binär wie Joystick ->
      ;1-links 2-rechts 3-hoch 4-runter, also bit 0&1 für re+li oder bit
4&2 für re&ru etc
              .he 0a 03 00 00 02 03 03 03 01
              .he 0c 00 03 00 00 00 00 00 00
              .he 0c 00 00 09 00 00 00 03 09
              .he 0c 00 00 0e 03 03 0b 00 0c
              .he 0c 00 00 05 00 00 0c 00 0c
              .he 06 03 03 00 00 00 06 03 05
```

It’s a simple binary code for each tile, we can walk:

0001 := left
0010 := right
0100 := up
1000 := down

$0a := 1010 := down and right
$0f := 1111 := all possible directions

There are some limitations:

Up and down is always straight - right and left can be diagonal, too.

When we move right or left, the routine first tests in straight direction. If there is no tile, it tests, if there is a tile in the upper line, when not we go to the lower line.

02 03 03 01   ok

```
02 00 00 00                 00 00 00 01
00 03 03 00     ok          00 03 03 00     ok
00 00 00 01                 02 00 00 00

02 00 00 00                 00 00 00 01
00 03 03 00     wrong       00 03 03 00     wrong
02 00 00 01                 02 00 00 01
```

These two are wrong, as you will always land on top tile.

```
08 00 00 00
0e 03 03 01     ok
04 00 00 00


0a 00 00 01
0e 03 03 00     wrong
04 00 00 00
```

Here we have 2 ways to land on the same tile.

Please remember that the direction the players has moved from is blocked to move back. Start tile (when direct landed) and move free in fortune wheel differ here. Warp tiles after warp don't have a moved from, too.

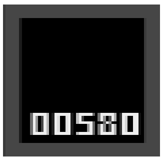
Next part is the definition of the categories of the plots:

```
l_kategorien
        .he 02 01 00 00 00 00 03 03 00
        .he 00 00 03 00 00 00 00 00 00
        .he 03 00 00 00 00 00 00 01 03
        .he 02 00 00 03 02 02 00 00 02
        .he 01 00 00 00 00 00 02 00 00
        .he 00 03 02 00 00 00 02 02 00
```

There are 3 different categories possible. 01 is the lowest one, 03 the highest.

cat 1	cat 2	cat 3
rent is 10% of worth	rent is 12% of worth	rent is 15% of worth
cat factor: 3	cat factor: 5	cat factor: 11

Maximum investment is calculated: cat factor x start price x plots in same area - start price

For this plot in level 1: 

- rent only this owned: 87

- investment only this owned: 11 x 580 - 580 = 5800

Now we set the areas, the tiles belong to:

```
l_bereiche  ;Bereiche in denen Miete teurer werden könnte
        .he 02 02 00 00 00 00 07 07 00
        .he 00 00 02 00 00 00 00 00 00
        .he 01 00 00 00 00 00 00 06 06
        .he 01 00 00 03 03 03 00 00 06
        .he 01 00 00 00 00 00 05 00 00
        .he 00 04 04 00 00 00 05 05 00
```

Areas are not limited. Every tile can have its own area. But owning some tiles within same area will have some more improvements. You can invest more money and you will get more rent.

Possible investment is calculated by the same formula as above. The rent is calculated with this neighbor factor (normal rent x neighbors), too. Owning another plot in the same area as the plot shown above results in double rent -> 87x2=174, possible maximum investment is then 12180.