Z-100 LifeLine
Since 1989

March 2022

#WEB

This article was first published in issue #131, March 2020
~~~~~~

ZBASIC CAKE.BAS

by Steven W. Vagts
Editor, "Z-100 LifeLine"

ZBASIC CAKE.BAS

The ZBASIC CAKE.BAS program was mostly a quick program to make issue #130 a bit more special. It was actually my wife's idea. However, it felt good to get back into ZBASIC and I've always been impressed with the graphics capability of such an early application. Perhaps it is nothing compared to today's standards, but in 1982?

If you took the time to play with it, I hope you found it interesting. If you no longer have ZBASIC capability, I understand, but hope that you will follow along with my explanation and reasoning. I'm sure you can probably find better ways to accomplish the same graphics affects, but that is one of the things that I like about ZBASIC - there are usually several ways to accomplish something, and usually there are tradeoffs on which way you may choose to go.

So let's discuss some of the more interesting aspects of this program.

First off, I wish everyone would include a Title block in their program. In addition to the Title, it should include the author & perhaps contact info & the date, but most importantly, what computer and version of the programming language is being used. BASIC programs especially seem to lack this important piece of information, and there are so many different versions, most of which have certain programming quirks that a person must figure out. Will it run with BASIC-80, QBASIC, GWBASIC, BASIC-80, etc.?

This CAKE.BAS program uses the H-19 ESCape codes and graphics characters, so it may work on the H-8 or H-88/89/90 using BASIC-80 or something similar. It will NOT work with anything later or some form of the PC-clone BASICs.

Following the Title block, I like to set up the various equates to use the ESCape codes that I will be using. So, generally starting at line 100, I like to define the ESCape character, E$, and then the ESCape codes to turn ON/OFF Reverse Video, Graphics Mode, sometimes even Colors to be used. I also like to initialize the ZBASIC RANDOMIZE function, in this case with line 130.

Looking in the ZBASIC manual, I could find no mention of RANDOMIZE TIME/DATE. So, I am not sure where I picked up this statement.

The reference manual only talks about the <expression> which is used as a random number seed value, and the examples simply show a number. But if you use the same number each time, the random number generated has the same number every time, and you have to remember to give a different number each time the program is run.

Just using RANDOMIZE TIME will not work; it generally gives the error, "Overflow in 130". For example, when I used the ZBASIC command PRINT TIME, DATE, and TIME/DATE, it gave:

```
       61279     16      3829.938
```

The Random Number Seed is limited to -32768 to +32768, hence the overflow error.

If you have not already done so, I recommend adding a note to the RANDOMIZE Statement of the Reference Guide, page 10.143 of the ZBASIC manual, "For a suitable Random Number Generator, use the statement **'RANDOMIZE TIME/DATE'**".

Next, the COLOR statement is obvious; it sets the screen colors. COLOR 1,3 sets Blue on a Cyan background.
~~~~~~

But the LINE statement of line 1010, could use some explanation. The LINE statement takes the form:

```
    LINE [(X1,Y1)]-(X2,Y2) [,[attribute]][,b[f]]

       Where:
          X1 and X2 are a column position
          Y1 and Y2 are a row position
          Attribute is a screen color
          B[f] is a fill background color
```

It permits the drawing of lines in absolute and relative locations on the screen.

LINE is the most powerful of the graphics statements. It is so important to my program that I will repeat the reference guide, page 10.90, here verbatim, with a few of my comments interspersed.

It allows a group of pixels to be controlled with a single statement.

A Pixel is the smallest point that can be plotted on the screen. If you get up close and personal with your Z-100 screen or display, you can make out the distinct dots of light that make up a character. Each of these dots of light is a pixel. The ZBASIC screen can display 25 lines of characters, each 80 characters long. This gives us a screen 640 (80x8) pixels long x 225 (25x9) pixels tall.

The simplest form of LINE is:

```
    LINE - (X2,Y2)
```

This will draw from the last point to the point (X2,Y2) in the foreground attribute.

We can include a starting point also:

```
    LINE (0,0) - 639,224)
```

This will draw a diagonal line down the screen.

The statement:

```
    LINE (0,100)-(639,100)
```

will draw a horizontal bar across the screen, 100 pixels down from the top line of pixels (about mid-screen).

We can append a color argument to draw the line in green, which is color two:

```
     LINE (10,10)-(20,20),2
```

If we used a RND Function, we could make the line appear anywhere on the screen in any random color.

The RND Function takes the form RND(X) and returns a random number between 0 and 1. The same sequence of random numbers is generated each time the program is run, unless the random number generator is reseeded using the RANDOMIZE statement discussed above.

However, X<0 always restarts the same sequence for any given X. X=0 repeats the last number generated. X>0 or X omitted generates the next random number in the sequence.

For example, the statement:

```
     PRINT INT(RND*100)
```

will print a number between 0 and 100. The INT function is used to restrict us to whole numbers (drops any decimal amount).

So, getting back to displaying our random lines and colors, we can use the program:

```
    10 CLS
    20 LINE -(RND*639,RND*224),RND*7
    30 GOTO 20
```

to draw lines forever on the screen using a random color for each.

The final optional argument to LINE is ",b" for a box, or ",bf" for a filled box. The syntax indicates that we can leave out the attribute argument and include the final argument as follows:

```
    LINE (0,0)-(100,100),,b
```

will draw a box in the foreground attribute.

Or:

```
    LINE (0,0)-(200,200),2,bf
```

will draw a filled box with color attribute 2 (Green).

The ",b" tells BASIC to draw a rectangle with the points (X1,Y1) and (X2,Y2) as opposite corners. This avoids giving the four separate LINE commands:

```
    LINE (X1,Y1)-(X2,Y2)
    LINE (X1,Y1)-(X1,Y2)
    LINE (X2,Y1)-(X2,Y2)
    LINE (X1,Y2)-(X2,Y2)
```

which perform the equivalent function.

The ",bf" means draw the same rectangle as ",b", but also fill in the interior of the box with the selected color attribute.

When out of range coordinates are given in the LINE command, the coordinate which is out of range is given the closest legal value. In other words, negative values become zero, Y values greater than 224 become 224 and X values greater than 639 become 639.

So, in our Cake program, the statement:

```
    1010 LINE (30,120)-(620,207),3,BF
```

draws our cake outline and fills in with Cyan.

To draw in our Cake message, we have the 2000 series of statements. The color of choice is set by COLOR 1,3, which is Blue on Cyan background.

The LOCATE statement takes the form:

```
    LOCATE [row],[col][,[cursor]]

       Where:
          Row is the screen line number between 1
          and 25 (not to be confused with pixel
          locations used elsewhere)
          Col is the screen column number between 1
          and 80.
          Cursor is set to indicate if it is visible
          or not. Zero is OFF, non-zero is ON.
```

The LOCATE statement moves the cursor to the desired screen character position. Subsequent PRINT statements begin placing characters at this position. Optionally, it may be used to turn the cursor ON or OFF.

So, our 2000 series statements place the cursor at the desired location for each character to be printed on the screen. The F$ puts us in Z-100 Graphics mode. The series of graphics letters is used to print our message in characters 3 lines high across the front of the cake.

When we are done, line #2075 is used to locate to Home, turn OFF graphics and reset our screen color to the default white on black background.

Now, it gets more interesting.

We want to fire rockets that explode in the air in the form of fireworks! At a fireworks display, we generally see a rocket fire into the air, it explodes with a POP, and then the remnants drift down until they burn out.
The series of statements beginning at line number 2100 sets us up for the first.

GOSUB 6000 is just a simple routine to set a random color for the rocket. You may notice that I had to add +1 to the equation. It seems that when INT drops the decimal, the number never gets to show 7 (white), and I did not want to show black on our black background. The simple fix was just to add one.

GOSUB 5000 was just to place a time delay for the rocket trail, the explosion, and the falling embers.

Another consideration was that we could use a random generated number and then an IF...THEN GOTO statement to fire each rocket from a random location on the cake. However, I found that when I was done, there was enough randomness that I did not need to make it any more complicated.

So I chose to fire the rockets from any letter that may look like a rocket launcher - ‘Y’, ‘H’, and ‘I’.

Two more comments of note, each rocket must be fired twice, once with the random color, and again at the default background color. This erases the track of the rocket from the screen.

We must also note the location at the end of the rocket’s track. We also need to convert the screen position numbers! The LINE command uses screen PIXEL positions, while the LOCATE command uses screen CHARACTER positions.

If you choose the rocket positions correctly (divisible by 9 and 8), the numbers are converted in line number 7010 to give you whole numbers for the explosion LOCATE command. However, you could easily just use INT to drop the decimal part of the location.

The last interesting element of the program is the 7000 series of statements to create the explosions.

As we did with the rocket trails, we need to draw the initial explosion, erase it with the next level of the explosion, erase that with the next level of the explosion, etc., until the last remnants of the explosion are deleted, for 5 levels. I liked the final affect.

I am sure there may be other ways to do this but this worked to my satisfaction. Adjust the time delay loop to adjust the timing if you wish.

That is about it. I hope you enjoyed the program. I found I enjoyed dusting off ZBASIC for this quick effort. Happy 100th issue to me.

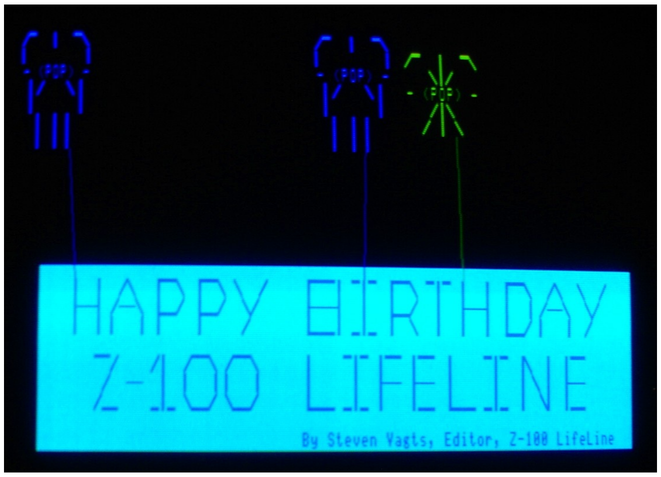
Note: For the pictures only, I had to make some changes to the program to leave the fireworks displayed.

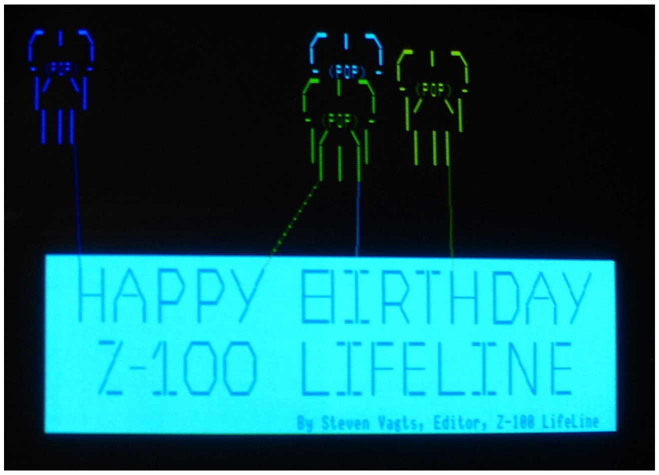
If you have any questions or comments, please email me at:
z100lifeline@swvagts.com

Cheers,

Steven W. Vagts

[illegible]





CAKE.BAS is a ZBASIC program to celebrate the 100th issue of the Z-100 LifeLine with Steven Vagts as Editor

```
10   CLS
20   REM  -----------------------------------------------------------------------------
30   REM      CAKE.BAS   -   100th Issue with Vagts as Editor
40   REM                          By Steven Vagts, Editor, Z-100 LifeLine
50   REM                             211 Sean Way, Hendersonville, NC 28792
60   REM      ZBASIC program for the H/Z-100             January 15, 2020
70   REM  -----------------------------------------------------------------------------

100  REM  Set ESCape Codes
110  E$=CHR$(27): F$=E$+”F”: G$=E$+”G”: REM  E$=ESCape, F$=Graphics ON, G$=OFF
120  P$=E$+”p”: Q$=E$+”q”: REM  P$=Reverse Video ON, Q$=OFF
130  RANDOMIZE TIME/DATE

1000 REM  Write Cake Outline and fill in Cyan
1010 LINE (30,120)-(620,207),3,BF

2000 COLOR 1,3: REM Write Cake Message to screen in Blue on Cyan
2005 REM  You must get every space & character correct, as shown.
2010 LOCATE 15,9:   PRINT F$+”`  `  xy  }zzy }zzy y  x”
2015 LOCATE 15,37: PRINT ”}zzz` z`z }zzy z`z `   ` }zzy  xy  y  x”
2020 LOCATE 16,9:   PRINT ”vaat xaay }{{x }{{x  yx”
2025 LOCATE 16,37: PRINT ”}aaat  `  }{{x  `  vaat }  } xaay  yx”
2030 LOCATE 17,9:   PRINT ”`  ` |  } }    }     x”
2035 LOCATE 17,37: PRINT ”}{{{` {`{ }  y  `  `  ` }{{x |  }  x”
2040 LOCATE 19,12: PRINT ”zzx    x`  xzzy xzzy”
2045 LOCATE 19,37: PRINT ”}    z`z `zz `zz }    z`z }y  | `zz”
2050 LOCATE 20,12: PRINT ” x  aa  `  |  } |  }”
2055 LOCATE 20,37: PRINT ”}      `  vaa vaa }     `  } y | vaa”
2060 LOCATE 21,12: PRINT ”x{{    {`{ y{{x y{{x”
2065 LOCATE 21,37: PRINT ”}{{{ {`{ `   `{{ }{{{ {`{ }  y| `{{”
2070 LOCATE 23,37: PRINT G$+”By Steven Vagts, Editor, Z-100 LifeLine”
2075 LOCATE 1,1:     PRINT G$: COLOR 7,0

2100 REM  Draw a random color rocket firework.
2110 GOSUB 6000: LINE (348,126)-(352,27),C
2120 GOSUB 5000: LINE (348,126)-(352,27),0: YC=352: XC=27
2130 GOSUB 7000
2140 REM  Note I left the rocket trail in the cake to see where it is launched.

2200 REM  Draw another rocket...
2210 GOSUB 6000: LINE (67,126)-(56,27),C
2220 GOSUB 5000: LINE (67,126)-(56,27),0: YC=56: XC=27
2230 GOSUB 7000

2300 REM  Draw another rocket...
2310 GOSUB 6000: LINE (444,126)-(440,36),C
2320 GOSUB 5000: LINE (444,126)-(440,36),0: YC=440: XC=36
2330 GOSUB 7000

2400 REM  Draw another rocket...
2410 GOSUB 6000: LINE (254,126)-(344,54),C
2420 GOSUB 5000: LINE (254,126)-(344,54),0: YC=344: XC=54
2430 GOSUB 7000

2500 REM  Draw another rocket...
2510 GOSUB 6000: LINE (591,126)-(600,48),C
2520 GOSUB 5000: LINE (591,126)-(600,48),0: YC=600: XC=48
2530 GOSUB 7000

2600 REM  Draw another rocket...
2610 GOSUB 6000: LINE (469,126)-(480,27),C
2620 GOSUB 5000: LINE (469,126)-(480,27),0: YC=480: XC=27
2630 GOSUB 7000

2700 REM  Draw another rocket...
2710 GOSUB 6000: LINE (223,126)-(136,36),C
2720 GOSUB 5000: LINE (223,126)-(136,36),0: YC=136: XC=36
2730 GOSUB 7000

2800 REM  Draw another rocket...
2810 GOSUB 6000: LINE (560,126)-(304,27),C
2820 GOSUB 5000: LINE (560,126)-(304,27),0: YC=304: XC=27
2830 GOSUB 7000

2900 REM  Draw another rocket...
2910 GOSUB 6000: LINE (91,126)-(96,63),C
2920 GOSUB 5000: LINE (91,126)-(96,63),0: YC=96: XC=63
2930 GOSUB 7000

4980 COLOR 7,0
4990 END

5000 REM  Time Delay
5010 FOR I=1 TO 50: NEXT I
5010 RETURN

6000 REM  Get a random color.
6010 C=INT(RND*7)+1
6020 RETURN

7000 REM  Create Exploding Firework, make same color as rocket.
7010 REM  Convert XC=RowCoord & YC=ColCoord from LINE Statement.
7020 X=XC/9: Y=YC/8:  REM  X & Y must be whole numbers, no remainder.
7030 COLOR C,0:  REM  Recover color of rocket.
7040 LOCATE X-1,Y-3:  PRINT ”\ | /”
7050 LOCATE X,Y-5:     PRINT ”- (POP) -”
7060 LOCATE X+1,Y-3: PRINT ”/ | \”: GOSUB 5000

7100 REM  Overwrite POP with expanding graphics.
7110 LOCATE X-2,Y-5:  PRINT F$+” zy ` xz ”
7120 LOCATE X-1,Y-5:  PRINT ”   y`x   ”
7130 LOCATE X,Y-5:     PRINT ” aa   aa ”
7140 LOCATE X+1,Y-5: PRINT ”   x`y   ”
7150 LOCATE X+2,Y-5: PRINT ”  x ` y  ”+G$: GOSUB 5000

7200 REM  Overwrite last set with expanding explosion.
7210 LOCATE X-2,Y-5:  PRINT F$+”xz  `  zy”
7220 LOCATE X-1,Y-5:  PRINT ”|       }”
7230 LOCATE X,Y-5:     PRINT ” x     y ”
7240 LOCATE X+1,Y-5: PRINT ” |x   y} ”
7250 LOCATE X+2,Y-5: PRINT ” || ` }} ”
7260 LOCATE X+3,Y-5: PRINT ”  | ` }  ”+G$: GOSUB 5000

7300 REM  Overwrite last set with expanding explosion.
7310 LOCATE X-2,Y-5:  PRINT F$+”         ”
7320 LOCATE X-1,Y-5:  PRINT ”         ”
7330 LOCATE X,Y-5:     PRINT ”         ”
7340 LOCATE X+1,Y-5: PRINT ”         ”
7350 LOCATE X+2,Y-5: PRINT ” |     } ”
7360 LOCATE X+3,Y-5: PRINT ”  | ` |  ”
7370 LOCATE X+4,Y-5: PRINT ”  | ` |  ”+G$: GOSUB 5000

7400 REM  Overwrite last remnants of explosion.
7410 LOCATE X+2,Y-5: PRINT ”         ”
7420 LOCATE X+3,Y-5: PRINT ”         ”
7430 LOCATE X+4,Y-5: PRINT ”         ”

7500 RETURN
```