What is code?

14	Sinus	Tangens	Secans
31	2506616	2589280	10329781
32	2509432	2592384	10330559
33	2512248	2595488	10331339
34	2515063	2598593	10332119
35	2517879	2601699	10332901
36	2520694	2604805	10333683
37	2523508	2607911	10334467
38	2526323	2611018	10335251
39	2529137	2614126	10336037
40	2531952	2617234	10336823
41	2534766	2620342	10337611
42	2537579	2623451	10338399
43	2540393	2626560	10339188
44	2543206	2629670	10339979
45	2546019	2632780	10340770
46	2548832	2635891	10341563
47	2551645	2639002	10342356
48	2554458	2642114	10343151
49	2557270	2645226	10343946
50	2560082	2648339	10344743
51	2562894	2651452	10345540
52	2565705	2654566	10346338
53	2568517	2657680	10347138
54	2571328	2660794	10347938
55	2574139	2663909	10348740
56	2576950	2667025	10349542
57	2579760	2670141	10350346
58	2582570	2673257	10351150
59	2585381	2676374	10351955
60	2588190	2679492	10352762

61

	Sinus	Tangens	Secans
29	9680748	38020782	39894421
28	9680018	38574537	39849654
27	9679288	38528396	39804991
26	9678557	38482358	39760431
25	9677825	38436424	39715975
24	9677092	38390591	39671621
23	9676358	38344861	39627369
22	9675624	38299233	39583219
21	9674888	38253707	39539171
20	9674152	38208281	39495224
19	9673415	38162957	39451379
18	9672678	38117733	39407633
17	9671939	38072609	39363988
16	9671200	38027585	39320443
15	9670459	37982661	39276997
14	9669718	37937835	39233651
13	9668977	37893109	39190403
12	9668234	37848481	39147254
11	9667490	37803951	39104203
10	9666746	37759519	39061250
9	9666001	37715185	39018395
8	9665255	37670947	38975637
7	9664508	37626807	38932976
6	9663761	37582763	38890411
5	9663012	37538815	38847943
4	9662263	37494963	38805570
3	9661513	37451207	38763293
2	9660762	37407546	38721112
1	9660011	37363980	38679025
0	9659258	37320508	38637033
75			

C
C
C
I
a
l
l
L
P
E
O
M

Diagram for the computation by the Engine of the Numbers of Bernoulli. See Note G. (page 722 *et seq.*)

Number of Operation.	Nature of Operation.	Variables acted upon.	Variables receiving results.	Indication of change in the value on any Variable.	Statement of Results.	Data.			Working Variables.										Result Variables.			
						$^1V_1$ ○ 0 0 1 [1]	$^1V_2$ ○ 0 0 2 [2]	$^1V_3$ ○ 0 0 4 [n]	$^0V_4$ ○ 0 0 0 [ ]	$^0V_5$ ○ 0 0 0 [ ]	$^0V_6$ ○ 0 0 0 [ ]	$^0V_7$ ○ 0 0 0 [ ]	$^0V_8$ ○ 0 0 0 [ ]	$^0V_9$ ○ 0 0 0 [ ]	$^0V_{10}$ ○ 0 0 0 [ ]	$^0V_{11}$ ○ 0 0 0 0 [ ]	$^0V_{12}$ ○ 0 0 0 0 [ ]	$^0V_{13}$ ○ 0 0 0 0 [ ]	$^1V_{21}$ ○ B_1 in a decimal fraction. [B_1]	$^1V_{22}$ ○ B_2 in a decimal fraction. [B_2]	$^1V_{23}$ ○ B_3 in a decimal fraction. [B_3]	$^0V_{24}$ ○ 0 0 0 [B_7]
1	×	$^1V_2 \times {}^1V_3$	$^1V_4, {}^1V_5, {}^1V_6$	$\begin{cases} {}^1V_2 = {}^1V_2 \\ {}^1V_3 = {}^1V_3 \end{cases}$	$= 2n$	...	2	n	$2n$	$2n$	$2n$											
2	−	$^1V_4 - {}^1V_1$	$^2V_4$	$\begin{cases} {}^1V_4 = {}^2V_4 \\ {}^1V_1 = {}^1V_1 \end{cases}$	$= 2n - 1$	1	...	...	$2n-1$													
3	+	$^1V_5 + {}^1V_1$	$^2V_5$	$\begin{cases} {}^1V_5 = {}^2V_5 \\ {}^1V_1 = {}^1V_1 \end{cases}$	$= 2n + 1$	1	...	...	...	$2n+1$												
4	÷	$^2V_5 \div {}^2V_4$	$^1V_{11}$	$\begin{cases} {}^2V_5 = {}^0V_5 \\ {}^2V_4 = {}^0V_4 \end{cases}$	$= \dfrac{2n-1}{2n+1}$	...	...	...	0	0	...	...	...	...	...	$\dfrac{2n-1}{2n+1}$						
5	÷	$^1V_{11} \div {}^1V_2$	$^2V_{11}$	$\begin{cases} {}^1V_{11} = {}^2V_{11} \\ {}^1V_2 = {}^1V_2 \end{cases}$	$= \dfrac{1}{2} \cdot \dfrac{2n-1}{2n+1}$	...	2	...	...	...	...	...	...	...	...	$\dfrac{1}{2} \cdot \dfrac{2n-1}{2n+1}$						
6	−	$^0V_{12} - {}^2V_{11}$	$^1V_{12}$	$\begin{cases} {}^2V_{11} = {}^0V_{11} \\ {}^0V_{12} = {}^1V_{12} \end{cases}$	$= -\dfrac{1}{2} \cdot \dfrac{2n-1}{2n+1} = A_0$	...	...	...	...	...	...	...	...	...	...	0	.....	$-\dfrac{1}{2} \cdot \dfrac{2n-1}{2n+1} = A_0$				
7	−	$^1V_3 - {}^1V_1$	$^2V_{10}$	$\begin{cases} {}^1V_3 = {}^1V_3 \\ {}^1V_1 = {}^1V_1 \end{cases}$	$= n - 1\ (= 3)$	1	...	n	...	...	...	...	...	...	$n-1$							
8	+	$^1V_2 + {}^0V_7$	$^1V_7$	$\begin{cases} {}^1V_2 = {}^1V_2 \\ {}^0V_7 = {}^1V_7 \end{cases}$	$= 2 + 0 = 2$	...	2	...	...	...	...	2										
9	+	$^1V_6 + {}^1V_7$	$^2V_{11}$	$\begin{cases} {}^1V_6 = {}^1V_6 \\ {}^0V_{11} = {}^2V_{11} \end{cases}$	$= \dfrac{2n}{2} = A_1$	...	...	...	...	...	$2n$	2	...	...	...	$\dfrac{2n}{2} = A_1$						
10	×	$^1V_{21} \times {}^2V_{11}$	$^1V_{12}$	$\begin{cases} {}^1V_{21} = {}^1V_{21} \\ {}^2V_{11} = {}^3V_{11} \end{cases}$	$= B_1 \cdot \dfrac{2n}{2} = B_1 A_1$	...	...	...	...	...	...	...	...	...	...	$\dfrac{2n}{2} = A_1$	$B_1 \cdot \dfrac{2n}{2} = B_1 A_1$	.....	B_1			
11	+	$^1V_{12} + {}^1V_{13}$	$^2V_{13}$	$\begin{cases} {}^1V_{12} = {}^0V_{12} \\ {}^1V_{13} = {}^2V_{13} \end{cases}$	$= -\dfrac{1}{2} \cdot \dfrac{2n-1}{2n+1} + B_1 \cdot \dfrac{2n}{2}$ ...	...	...	...	...	...	...	...	...	...	...	.....	0	$\left\{ -\dfrac{1}{2} \cdot \dfrac{2n-1}{2n+1} + B_1 \cdot \dfrac{2n}{2} \right\}$				
12	−	$^1V_{10} - {}^1V_1$	$^2V_{10}$	$\begin{cases} {}^1V_{10} = {}^2V_{10} \\ {}^1V_1 = {}^1V_1 \end{cases}$	$= n - 2\ (= 2)$	1	...	...	...	...	...	...	...	...	$n-2$							
13	−	$^1V_6 - {}^1V_1$	$^2V_6$	$\begin{cases} {}^1V_6 = {}^2V_6 \\ {}^1V_1 = {}^1V_1 \end{cases}$	$= 2n - 1$	1	...	...	...	...	$2n-1$											
14	+	$^1V_1 + {}^1V_7$	$^2V_7$	$\begin{cases} {}^1V_1 = {}^1V_1 \\ {}^1V_7 = {}^2V_7 \end{cases}$	$= 2 + 1 = 3$	1	...	...	...	...	...	3										
15	+	$^2V_6 + {}^2V_7$	$^1V_8$	$\begin{cases} {}^2V_6 = {}^2V_6 \\ {}^2V_7 = {}^2V_7 \end{cases}$	$= \dfrac{2n-1}{3}$	...	...	...	...	...	$2n-1$	3	$\dfrac{2n-1}{3}$									
16	×	$^1V_8 \times {}^2V_{11}$	$^4V_{11}$	$\begin{cases} {}^1V_8 = {}^0V_8 \\ {}^2V_{11} = {}^4V_{11} \end{cases}$	$= \dfrac{2n}{2} \cdot \dfrac{2n-1}{3}$	...	...	...	...	...	...	...	0	...	...	$\dfrac{2n}{2} \cdot \dfrac{2n-1}{3}$						
17	−	$^2V_6 - {}^1V_1$	$^3V_6$	$\begin{cases} {}^2V_6 = {}^3V_6 \\ {}^1V_1 = {}^1V_1 \end{cases}$	$= 2n - 2$	1	...	...	...	...	$2n-2$											
18	+	$^1V_1 + {}^2V_7$	$^3V_7$	$\begin{cases} {}^2V_7 = {}^3V_7 \\ {}^1V_1 = {}^1V_1 \end{cases}$	$= 3 + 1 = 4$	1	...	...	...	...	...	4										
19	÷	$^2V_6 \div {}^3V_7$	$^1V_9$	$\begin{cases} {}^3V_6 = {}^3V_6 \\ {}^3V_7 = {}^3V_7 \end{cases}$	$= \dfrac{2n-2}{4}$	...	...	...	...	...	$2n-2$	4	...	$\dfrac{2n-2}{4}$	...	$\left\{ \dfrac{2n}{2} \cdot \dfrac{2n-1}{3} \cdot \dfrac{2n-2}{3} \right\} = A_2$							
20	×	$^1V_9 \times {}^4V_{11}$	$^5V_{11}$	$\begin{cases} {}^1V_9 = {}^0V_9 \\ {}^4V_{11} = {}^5V_{11} \end{cases}$	$= \dfrac{2n}{2} \cdot \dfrac{2n-1}{3} \cdot \dfrac{2n-2}{4} = A_2$	...	...	...	...	...	...	...	...	0								
21	×	$^1V_{22} \times {}^3V_{11}$	$^2V_{12}$	$\begin{cases} {}^1V_{22} = {}^1V_{22} \\ {}^0V_{12} = {}^2V_{12} \end{cases}$	$= B_2 \cdot \dfrac{2n}{2} \cdot \dfrac{2n-1}{3} \cdot \dfrac{2n-2}{3} = B_2 A_2$	...	...	...	...	...	...	...	...	...	...	0	$B_2 A_2$	.....	.....	B_2		
22	+	$^2V_{12} + {}^2V_{13}$	$^3V_{13}$	$\begin{cases} {}^2V_{12} = {}^0V_{12} \\ {}^2V_{13} = {}^3V_{13} \end{cases}$	$= A_0 + B_1 A_1 + B_2 A_2$	...	...	...	...	...	...	...	...	...	...	.....	0	$\left\{ A_0 + B_1 A_1 + B_2 A_2 \right\}$				
23	−	$^2V_{10} - {}^1V_1$	$^3V_{10}$	$\begin{cases} {}^2V_{10} = {}^3V_{10} \\ {}^1V_1 = {}^1V_1 \end{cases}$	$= n - 3\ (= 1)$	1	...	...	...	...	...	...	...	...	$n-3$							
Here follows a repetition of Operations thirteen to twenty-three.																						
24	+	$^4V_{13} + {}^0V_{24}$	$^1V_{24}$	$\begin{cases} {}^4V_{13} = {}^0V_{13} \\ {}^0V_{24} = {}^1V_{24} \end{cases}$	$= B_7$	...	...	...	...	...	...	...	...	...	...	.....	.....	.....	.....	.....	.....	B_7
25	+	$^1V_1 + {}^1V_3$	$^1V_3$	$\begin{cases} {}^1V_1 = {}^1V_1 \\ {}^1V_3 = {}^1V_3 \\ {}^4V_6 = {}^0V_6 \\ {}^4V_7 = {}^0V_7 \end{cases}$	$= n + 1 = 4 + 1 = 5$ ..... by a Variable-card. by a Variable card.	1	...	$n+1$	...	...	0	0										

BOMBE

Q W E R T Z U I O
A S D F H J K
F Y X C V B N M L

“It is possible to invent a single machine which can be used to compute any computable sequence.”

ON
OFF

ON
OFF

I_{ce}
... it allows for
a larger current
to flow here
c
b
I_{be}
e
If a small current
flows here...

A
B
OR
Q
A
B
AND
Q
A
B
XOR
Q
A
B
NOR
Q
A
B
NAND
Q
A
NOT
$\overline{A}$

A
B
Cin
XOR
XOR
Sum
NAND
NAND
NAND
Cout
Full Adder

ON
OFF
ON
OFF
OFF
OFF
OFF
OFF
ON
NAND
XOR
AND
AND
OR
XOR
AND
XOR
AND
AND
NAND
OR
AND
NAND
AND
OR
XOR
NAND
AND
AND
OR
XOR
OR
NAND
XOR
XOR
AND
AND
XOR
OR
XOR
OR
XOR
XOR
AND
XOR
AND
OR
NAND
XOR
NAND
OR
XOR
NAND
OR
NAND
XOR
OR
XOR
NAND
AND
XOR
AND
AND
AND

C4004
N5832

DEC	HEX	BIN	DEC	HEX	BIN	DEC	HEX	BIN
0	00	00000000	43	2B	00101011	86	56	01010110
1	01	00000001	44	2C	00101100	87	57	01010111
2	02	00000010	45	2D	00101101	88	58	01011000
3	03	00000011	46	2E	00101110	89	59	01011001
4	04	00000100	47	2F	00101111	90	5A	01011010
5	05	00000101	48	30	00110000	91	5B	01011011
6	06	00000110	49	31	00110001	92	5C	01011100
7	07	00000111	50	32	00110010	93	5D	01011101
8	08	00001000	51	33	00110011	94	5E	01011110
9	09	00001001	52	34	00110100	95	5F	01011111
10	0A	00001010	53	35	00110101	96	60	01100000
11	0B	00001011	54	36	00110110	97	61	01100001
12	0C	00001100	55	37	00110111	98	62	01100010
13	0D	00001101	56	38	00111000	99	63	01100011
14	0E	00001110	57	39	00111001	100	64	01100100
15	0F	00001111	58	3A	00111010	101	65	01100101
16	10	00010000	59	3B	00111011	102	66	01100110
17	11	00010001	60	3C	00111100	103	67	01100111
18	12	00010010	61	3D	00111101	104	68	01101000
19	13	00010011	62	3E	00111110	105	69	01101001
20	14	00010100	63	3F	00111111	106	6A	01101010
21	15	00010101	64	40	01000000	107	6B	01101011
22	16	00010110	65	41	01000001	108	6C	01101100
23	17	00010111	66	42	01000010	109	6D	01101101
24	18	00011000	67	43	01000011	110	6E	01101110
25	19	00011001	68	44	01000100	111	6F	01101111
26	1A	00011010	69	45	01000101	112	70	01110000
27	1B	00011011	70	46	01000110	113	71	01110001
28	1C	00011100	71	47	01000111	114	72	01110010
29	1D	00011101	72	48	01001000	115	73	01110011
30	1E	00011110	73	49	01001001	116	74	01110100
31	1F	00011111	74	4A	01001010	117	75	01110101
32	20	00100000	75	4B	01001011	118	76	01110110
33	21	00100001	76	4C	01001100	119	77	01110111
34	22	00100010	77	4D	01001101	120	78	01111000
35	23	00100011	78	4E	01001110	121	79	01111001
36	24	00100100	79	4F	01001111	122	7A	01111010
37	25	00100101	80	50	01010000	123	7B	01111011
38	26	00100110	81	51	01010001	124	7C	01111100
39	27	00100111	82	52	01010010	125	7D	01111101
40	28	00101000	83	53	01010011	126	7E	01111110
41	29	00101001	84	54	01010100	127	7F	01111111
42	2A	00101010	85	55	01010101			

46684

Programming languages

```
C000                    ORG    ROM+$0000 BEGIN MONITOR
C000 8E 00 70  START    LDS    #STACK

               ****************************************
               * FUNCTION: INITA - Initialize ACIA
               * INPUT: none
               * OUTPUT: none
               * CALLS: none
               * DESTROYS: acc A

0013           RESETA  EQU    %00010011
0011           CTLREG  EQU    %00010001

C003 86 13     INITA   LDA A  #RESETA   RESET ACIA
C005 B7 80 04          STA A  ACIA
C008 86 11             LDA A  #CTLREG   SET 8 BITS AND 2 STOP
C00A B7 80 04          STA A  ACIA

C00D 7E C0 F1          JMP    SIGNON    GO TO START OF MONITOR

               ****************************************
               * FUNCTION: INCH - Input character
               * INPUT: none
               * OUTPUT: char in acc A
               * DESTROYS: acc A
               * CALLS: none
               * DESCRIPTION: Gets 1 character from terminal

C010 B6 80 04  INCH    LDA A  ACIA      GET STATUS
C013 47                ASR A            SHIFT RDRF FLAG INTO CARRY
C014 24 FA             BCC    INCH      RECIEVE NOT READY
C016 B6 80 05          LDA A  ACIA+1    GET CHAR
C019 84 7F             AND A  #$7F      MASK PARITY
C01B 7E C0 79          JMP    OUTCH     ECHO & RTS

               ****************************************
               * FUNCTION: INHEX - INPUT HEX DIGIT
               * INPUT: none
               * OUTPUT: Digit in acc A
               * CALLS: INCH
               * DESTROYS: acc A
               * Returns to monitor if not HEX input

C01E 8D F0     INHEX   BSR    INCH      GET A CHAR
C020 81 30             CMP A  #'0       ZERO
C022 2B 11             BMI    HEXERR    NOT HEX
C024 81 39             CMP A  #'9       NINE
C026 2F 0A             BLE    HEXRTS    GOOD HEX
C028 81 41             CMP A  #'A
C02A 2B 09             BMI    HEXERR    NOT HEX
C02C 81 46             CMP A  #'F
C02E 2E 05             BGT    HEXERR
C030 80 07             SUB A  #7        FIX A-F
C032 84 0F     HEXRTS  AND A  #$0F      CONVERT ASCII TO DIGIT
C034 39                RTS

C035 7E C0 AF  HEXERR  JMP    CTRL      RETURN TO CONTROL LOOP
```

As an example, the following program uses the Newton quotient to estimate $f'(x)$ for $f(s) = x^2$ at $x = 2$, for smaller and smaller values of h (the exact answer is 4).

```
REAL NQ, X, H
INTEGER I, N
X = 2
H = 1
PRINT*, "How many?"
READ*, N
PRINT "(A10, A12)", "H", "NQ"

DO I = 1, N
  NQ = (F(X + H) - F(X)) / H
  PRINT "(E10.4, F12.8)", H, NQ
  H = H / 10
END DO

CONTAINS
  FUNCTION F(X)
  REAL F, X
    F = X ** 2
  END FUNCTION F
END
```

Output:

```
         H          NQ
0.1000E+01  5.00000000
0.1000E00  4.09999371
0.1000E-01  4.01000977
0.1000E-02  4.00066423
0.1000E-03  3.99589586
0.1000E-04  4.00543261
0.1000E-05  3.81469774
```

The results show that the best h for this particular problem is about 10^{-5}. But for h much smaller than this the estimate becomes totally unreliable. Using double precision real kind improves the accuracy. Change `REAL` to `REAL(2)`:

```
         H          NQ
...
0.1000E-04  4.00001000
0.1000E-05  4.00000100
0.1000E-06  4.00000009
0.1000E-07  3.99999998
0.1000E-08  4.00000033
0.1000E-09  4.00000033
0.1000E-10  4.00000033
0.1000E-11  4.00035560
```

The best value is now about 10^{-8}.

Generally, the best h for a given problem can only be found by trial and error. Finding it constitutes a major problem of numerical analysis. This problem does not arise with numerical integration, because numbers are added to find the area, not subtracted.

```
EDIT       AGY0157.DEMO.SRCLIB(PROG45) - 01.01            Columns 00001 00072
Command ===>                                               Scroll ===> CSR
=COLS> ----+----1----+----2----+----3----+----4----+----5----+----6----+----7--
000007          WORKING-STORAGE SECTION.
000008         *-------------------------------------------------*
000009         *              DB2 STORAGE AREA - EMPLOYEE        *
000010         *-------------------------------------------------*
000011              EXEC SQL
000012                  DECLARE AGY0157.EMPLOYEE TABLE(
000013                  EMPID  INTEGER,
000014                  ENAME  CHAR(10),
000015                  SALARY DECIMAL(7,2),
000016                  JDATE  DATE)
000017              END-EXEC.
```

10 COLOR 1,0
20 CLS
30 FOR X = 1 TO 63
40 PRINT "HELLO WORLD"
50 COLOR X,0
60 NEXT X

```
599 REM POPPING SOUND
600 FOR S=15 TO 0 STEP -1
    :SOUND 0,15,0,S:SOUND
     1,16,0,S:NEXT S
610 SOUND 0,0,0,0:SOUND 1
    ,0,0,0:RETURN
999 REM CHECK FOR COLLISI
    ON
1000 PEK=PEEK(53261):IF P
     EK=0 THEN RETURN
1002 GOTO 1010
1005 FOR I=25 TO 10 STEP
     -5:SOUND 0,I,4,8:SOU
     ND 1,I+2,2,8:NEXT I:
     SOUND 0,0,0,0:SOUND
     1,0,0,0:RETURN
1009 REM DID BALLOON HIT
     HEAD?
1010 IF BY-BB<3 THEN PM$(
     H+BB,P0+BB)=B$:BB=BB
     -11:BAL=BAL-1:BOH=BO
     H+1:BAL(BOH)=C(I-1,B
     R/3):POP :GOTO 60
1019 REM DID ARM HIT BALL
     OON?
1020 IF STRIG(0)=0 THEN I
     F ABS(BY-178)<3 THEN
      1040
1030 A=A+((A>PP)-(A<=PP))
     *3:POKE 53249,A:POKE
      PC,1:RETURN
1040 P=A:IF HIT THEN P=PP
     +((P>PP)-(P<=PP))*3
1050 FOR K=BY TO BL STEP
     -3:GOSUB 500:SOUND 0
     ,K,10,8:PM$(P1+K,F+K
     )=B$:POKE 53249,P
1060 P=P+(((P>PP)-(P<=PP)
     )*(P>65 AND P<200)*(
     PEEK(53261)=0)):POKE
      PC,1
1070 NEXT K:SOUND 0,0,0,0
     :HIT=1:POP :GOTO 40
1499 REM MAN MISSED BALLO
     ON
1500 PM$(P1+BY,P1+BY+100)
     =N$
1510 PM$(PM,PB)=P$(1,37):
     PM$(P0+190,PB)=MDR$:
     POKE 53250,PP-8:FOR
     I=200 TO 0 STEP -10:
     SOUND 0,I,2,10:NEXT
     I
1520 SOUND 0,0,0,0:IF BOH
     <1 THEN 1550
1530 FOR I=158 TO BB STEP
      -11:FOR J=I TO 175
1540 PM$(P0+J,P0+J+12)=B$
     :NEXT J:PM$(P0+J,P0+
     J+12)=POB$:GOSUB 600
     :FOR K=1 TO 9:NEXT K
     :PM$(P0+J,P0+J+12)=N
     $:NEXT I
1550 FOR I=1 TO 100:NEXT
     I:POKE 53250,0:COLOR
      32:PLOT LF,23:SOUND
      0,0,0,0
1560 PM$(PT,PB)=P$:POKE 5
     3248,PP:BOH=0:BB=169
     :PF=120:RETURN
1999 REM POP BALLOONS AND
      TALLY POINTS
2000 IF BOH<1 THEN RETURN

2010 FOR I=1 TO BOH:PM$(P
     M+1,PD+1)=H$
2020 PM$(P0+158,PM)=POB$:
     GOSUB 600:FOR V=1 TO
      15:NEXT V:PM$(P0+15
     8,PM)=N$:BOH=BOH-1:P
     M$(PM,PB)=P$(100,137
     )
2030 SC=SC+BAL(I)*5*LVL:P
     OSITION 7,0:? #6;SC:
```

Player/missile graphics animate the player and balloons in the Atari version of "Balloon Crazy."

```
     IF SC>BONUS THEN LF=
     LF+1:BONUS=BONUS+100
     0*LVL:COLOR 72:PLOT
     LF,23
2040 IF BOH<1 THEN 2060
2050 FOR K=1 TO BOH:FOR J
     =158-K*11 TO 168-K*1
     1:PM$(P0+J,P0+J+12)=
     B$:NEXT J:NEXT K:NEX
     T I
2060 PM$(PT,PM)=N$:RETURN

2499 REM CHOOSE BALLOON T
     O FALL
2500 BR=3+INT(RND(0)*6)*3
     :FOR I=5 TO 2 STEP -
     1:GOSUB 500:LOCATE B
     R,I,BT:IF BT=32 THEN
      NEXT I:GOTO 2500
2510 POKE 705,PEEK(707+C(
     I-1,BR/3)):A=24*(BR/
     3+2):BL=32+I*8:POKE
     53249,A
2520 PM$(P1+BL,D+BL)=B$:C
     OLOR 32:PLOT BR,I
2530 BS=(C(I-1,BR/3)+2)/2
     :SL=INT((LVL/3+1)/2+
     BS):RETURN
2999 REM DRAW SCREEN
3000 RESTORE 3000:DIM F(3
     ),C(4,6),BAL(6):FOR
     I=1 TO 3:READ A:F(I)
     =A:NEXT I:DATA 0,32,
     128
3010 PP=120:POKE 53248,PP
     :BAL=24:LVL=LVL+1:IF
      LVL>10 THEN LVL=10
3020 POSITION 11,23:? #6;
     "[illegible]":LVL:FOR I=1
      TO LF:COLOR 72:PLOT
      I,23:NEXT I
3030 FOR X=3 TO 18 STEP 3
     :FOR Y=2 TO 5:A=INT(
     RND(0)*3)+1:COLOR 79
     +F(A):C(Y-1,X/3)=A:P
     LOT X,Y:NEXT Y:NEXT
     X:RETURN
3500 DIM T$(13):T$="BALLO
     ON [illegible]":POKE 708,1
     34:POKE 709,198:POKE
      710,86
3510 FOR I=4 TO 16:POSITI
     ON 1,I:? #6;T$(I-3,I
     -3):NEXT I
3520 POSITION 1,0:? #6;"s
     core:";SC
3530 RETURN
3999 REM SET UP VARIABLES
      AND STARTING DISPLA
     Y
4000 PP=120:BB=169:SC=0:L
     VL=0:LF=3:PC=53278:H
     =P0-12:PT=P0+69:PB=P
     0+206
4010 PM=PB-38:D=P1+12:F=D
     +3:BT=PB-10:PD=PM+23
     :BONUS=1000
4020 GRAPHICS 18:POKE 708
     ,134:POKE 709,198:PO
     KE 710,84:POKE 756,C
     HS/256
4030 POSITION 3,4:? #6;"B
     ALL[illegible]ON crazy!":POSIT
     ION 4,6:? #6;"[illegible]
     [illegible]"
4040 DIM S$(10):S$="Oo[illegible]Oo
     [illegible]Oo[illegible]O":FOR I=0 TO 10
      STEP 10:POSITION I,
     0:? #6;S$:POSITION I
     ,10:? #6;S$:NEXT I
4050 IF PEEK(53279)<>6 TH
     EN 4050
4060 RETURN
4100 POKE 53248,0:POKE 53
     249,0:GRAPHICS 17:PO
     KE 708,134:POKE 709,
     198:POKE 710,84
4110 POSITION 6,4:? #6;"g
     ame over":POSITION 8
     ,6:? #6;"SCORE:"
4120 POSITION 9,8:? #6;SC
     :POSITION 5,10:? #6;
     "[illegible]":POSITI
     ON 4,12:? #6;"[illegible]
     [illegible]"
4130 IF PEEK(53279)<>6 TH
     EN 4130
4140 CLR :GOTO 10
4499 REM REDEFINE CHARACT
     ERS
4500 GRAPHICS 17:POKE 559
     ,0:DIM C$(2):C$="OH"
4510 CHS=(PEEK(106)-8)*25
     6:CHO=57344
4520 IF PEEK(CHS+9)<>0 TH
     EN RETURN
4530 FOR I=0 TO 511:POKE
     CHS+I,PEEK(CHO+I):NE
     XT I
4540 RESTORE 4560:FOR I=1
      TO 2:CHP=CHS+(ASC(C
     $(I))-32)*8:FOR J=0
     TO 7:READ A:POKE CHP
     +J,A:NEXT J:NEXT I
4550 FOR I=32 TO 39:POKE
     CHS+I,255-PEEK(CHO+I
     ):NEXT I:RETURN
4560 DATA 28,58,125,125,1
     21,62,28,8
4570 DATA 0,16,56,124,56,
     60,56,0
4999 REM STE UP P/M
5000 DIM PM$(4096),P$(255
     ),N$(12),B$(15),POB$
     (12),MDL$(16),MDR$(1
     6),H$(22)
5010 FOR I=1 TO 12:N$(I)=
     CHR$(0):NEXT I
5020 A=ADR(PM$):PMB=INT(A
     /2048)*2048:IF PMB<A
      THEN PMB=PMB+2048
5030 S=PMB-A:POKE 54279,P
     MB/256:POKE 53277,3
5040 PM$=CHR$(0):PM$(4096
     )=CHR$(0):PM$(2)=PM$
     :P$=PM$
5050 P0=S+1024:POKE 704,5
     4:RESTORE 5070
5060 FOR I=100 TO 137:REA
     D A:P$(I)=CHR$(A):NE
     XT I:PM$(P0+69,P0+20
     6)=P$
5070 DATA 0,16,16,56,56,1
     24,124,56,56,60,60,5
     6,56,16,16,56,56,124
     ,124,186,186,185,185
     ,120,120,40,40,40,40
     ,40,40
```

?FD 100
?RT 90
?FD 100
?

Design By Numbers MIT Media Laboratory, Aesthetics + Computation Group, (C) 1999, Massachusetts Institute of Technology
100
80
60
40
20
0
0 20 40 60 80 100
// enter program
{
line 0 0 100 80
line 80 80 0 0
line 90 80 0 1
line 100 80 0 2
}

```
CANVAS = 500
SQUARESIZE = 158
NSQUARES = 50
SQUAREDIST = 6

width = NSQUARES * SQUAREDIST


size(CANVAS, CANVAS)

translate(CANVAS/2 - width / 2, CANVAS/2)

fill(1)
stroke(0)

startAngle = 45
endAngle = 45 + 180

for i in range(NSQUARES + 1):
    f = i / NSQUARES
    save()
    translate(i * SQUAREDIST)
    scale(0.7, 1)
    rotate(startAngle + f * (endAngle - startAngle))
    rect(-SQUARESIZE/2, -SQUARESIZE/2, SQUARESIZE, SQUARESIZE)
    restore()
```

gpuParticl...ple Debug | My Mac (64-bit) | Running gpuParticleSystemExampleDebug : gpuParticleSystemExample Debug

gpuParticleSystemExample
- openFrameworks-Info.plist
- Project.xcconfig
- src
  - main.cpp
  - ofApp.cpp
  - ofApp.h
- openFrameworks
- addons
- local_addons
- gpuParticleSyste...xampleDebug.app

gpuParticleSystemExample > src > ofApp.h > No Selection

```
/*
 *  summary:Example of how to use GPU for data processing. The data it's going to be stored
 *          on the color channels of the FBO's textures. In this case we are going to use just
 *          RED and GREEN channels on two textures. One for the position and the other one for
 *          the velocity. For updating the informacion of those textures we are going to use
 *          two FBO's for each type of information. This pair of FBO will pass the information
 *          from one to other in a techninc call PingPong.
 *          After updating this information, we are going to use the textures allocated on GPU memory
 *          for moving some vertex and then multiplied them in order to make little frames that hold
 *          a texture of a spark of light.
 */

#pragma once

#include "ofMain.h"

// Struct for doing PingPong quickly and easy
//
// Because on GPU you can't write over the texture that you are reading we are
// using to pair of ofFbo attached together on what we call pingPongBuffer
// Learn more about Ping-Pong at:
//
// http://www.comp.nus.edu/~ashwinna/docs/Ping
// http://www.seas.upenn.edu/~cis565/fbo.htm#
//
struct pingPongBuffer {
public:
    void allocate( int _width, int _height, i
        // Allocate
        for(int i = 0; i < 2; i++){
            FBOs[i].allocate(_width,_height, _
            FBOs[i].getTexture().setTextureMi
        }

        // Clean
        clear();
    }

    void swap(){
        std::swap(src,dst);
    }

    void clear(){
        for(int i = 0; i < 2; i++){
            FBOs[i].begin();
            ofClear(0,255);
            FBOs[i].end();
        }
    }

    ofFbo& operator[]( int n ){ return FBOs[n]
    ofFbo   *src = &FBOs[0];        // Source
    ofFbo   *dst = &FBOs[1];        // Destinat

private:
    ofFbo   FBOs[2];    // Real addresses of p
};


class ofApp : public ofBaseApp{
public:
    void setup();
    void update();
    void draw();

    void keyPressed  (int key);
```

Identity and Type
- Name: ofApp.h
- Type: Default - C Header
- Location: Relative to Project
- src/ofApp.h
- Full Path: /Users/danielpowers/Desktop/of_v0.9.3_osx_release/examples/gl/gpuParticleSystemExample/src/ofApp.h

On Demand Resource Tags

Add to a target to enable tagging

sketch_160312b | Processing 3.0

sketch_160312b

```
int numBalls = 250;

float[] xPositions;
float[] yPositions;

float[] xSpeeds;
float[] ySpeeds;

void setup() {
  size(500, 500);
  background(255);

  xPositions = new float[numBalls];
  yPositions = new float[numBalls];

  xSpeeds = new float[numBalls];
  ySpeeds = new float[numBalls];

  for (int i=0; i<numBalls; i++) {
    xPositions[i] = width / 2;
    yPositions[i] = height / 2;

    xSpeeds[i] = random(-5, 5);
    ySpeeds[i] = random(-5, 5);
  }
}

void draw() {
  noStroke();
  fill(255, 20);
  rect(0, 0, width, height);

  for (int i=0; i<numBalls; i++) {
    fill(0);
```

Save canceled.

Console Errors

sketch_160312b

Code and art

Unhappy Ready Made - Duchamp - 1919

CLOUD PIECE

Imagine the clouds dripping.
Dig a hole in your garden to
put them in.

1963 spring

Cloud Piece (from Grapefruit) - Yoko Ono - 1963

Flux Box 2 - 1967

Water Yam - George Brecht - 1963

Take an object.
Do something to it.
Do something else to it.
〃 〃 〃 〃 〃

Jasper Johns, Notebook, 1967

A 36 x 36 removal to the lathing or support wall of plaster or wallboard from a wall
Lawrence Weiner - 1968

http://ericdoeringer.com

Two Minutes of Spray Paint Directly on the Floor From a Standard Aerosol Spray Can - Lawrence Weiner

PROPOSAL FOR WALL DRAWING, INFORMATION SHOW

Within four adjacent squares,
each 4' by 4',
four draftsmen will be employed
at $4.00/hour
for four hours a day
and for four days to draw straight lines
4 inches long
using four different colored pencils;
9H black, red, yellow and blue.
Each draftsmen will use the same color throughout
the four day period,
working on a different square each day.

Wall Drawing #419

The wall is bordered and divided horizontally and vertically into four equal parts with a 6-inch (15 cm) black ink band. Each quarter has alternating parallel 6-inch (15 cm) bands of white and color ink bands. Upper left: gray; upper right: yellow; lower left: red; lower right: blue.

Wall Drawing #630

A wall is divided horizontally into two equal parts. Top: alternating horizontal black and white 8-inch (20 cm) bands. Bottom: alternating vertical black and white 8-inch (20 cm) bands.

arcs from four corners

from four sides

straight lines

not-straight lines

broken lines

Sol LeWItt - Wall Drawing #260

Sol LeWitt - Wall Drawing #260 - MOMA

PROPOSAL FOR WALL DRAWING, INFORMATION SHOW

Within four adjacent squares,
each 4' by 4',
four draftsmen will be employed
at $4.00/hour
for four hours a day
and for four days to draw straight lines
4 inches long
using four different colored pencils;
9H black, red, yellow and blue.
Each draftsmen will use the same color throughout
the four day period,
working on a different square each day.

Whirlwind computer - MIT - 1955

APS INSTRUCTIONAL COMPUTING CENTER

After Jean Auguste Ingres - 1963

After Durer - 1963

After Picasso - 1963

After Mondrian - 1963

On Growth and From - D'Arcy Thompson - 1963

b h g f e d c a
i
k
l
p
q m
r s
n
b h g f e d c a
i
k
l
p
q m
r s
n

aig c d e f h k b
aig c d e f h k b

Sine Landscape - Charles Csuri - 1967

Sine Wave Man - Charles Csuri - 1967

Random War - Charles Csuri - 1967

(1) Dead
(2) Wounded
(3) Missing
(4) Survivors
(5) One Hero for Each Side
(6) Medals for Valor
(7) Good Conduct
(8) Efficiency Medals

Random War - Charles Csuri - 1967

Computer composition with lines
A. Michael Noll - 1967

Composition with lines
Mondrian - 1917

Fig. 4 Picture pair as presented on separate sheets to subjects. The original microfilm and Mondrian photograph were copied xerographically and then reproduced so that both pictures were identical in quality. (© A. Michael Noll 1965 and © Rijkmuseum Kröller-Müller.)

Victory Boogie Woogie
Piet Mondrian - 1944

MVBW–1
Edwin Jakobs - 2013

Packer
Edwin Jakobs - 2012

Packer
Edwin Jakobs - 2012

Packer
Edwin Jakobs - 2012

Felder von Rechteck, Schrauffen Überlagert - Frieder Nake - 1965

Walk Through Raster - Frieder Nake - 1965

Frieder Nake - 1965

144 Trapèzes - Vera Molnar - 1974

The computer helps, but it does not 'do', does not 'design' or 'invent' anything

Vera Molnar

Scratch Code - Manfred Mohr - 1970 - 1975

Cubic Limit - Manfred Mohr - 1977

Cluster Phobia- Manfred Mohr - 1972

Scratch Code - Manfred Mohr - 1972

PROG. 21
MOHR 70

Digital graphic produced with a random number generator that determines length, direction (0°, 45°, 90°), line thickness, etc

My artistic goal is reached, when a finished work can visually dissociate itself from its logical content and convincingly stand as an independent abstract entity

Manfred Mohr

p-300b - Manfred Mohr - 1980

Cubic Limit - Manfred Mohr - 1974

E.A.T.

E.A.T. —The Story of Experiments in Art and Technology

NTT InterCommunication Center [ICC]

E.A.T.——芸術と技術の実験

NTTインターコミュニケーション・センター [ICC]

9 evenings: theatre & engineering
OCTOBER 13·14·15·16 18·19 21·22·23 8:30 P.M. $3 each evening
25th STREET ARMORY NYC TELEPHONE 689-3315
PERFORMANCES OF DANCE·MUSIC·FILM·TELEVISION·TECHNOLOGY BY CAGE·CHILDS·FÄHLSTROM·
HAY·HAY·PAXTON·RAINER·RAUSCHENBERG·TUDOR· WHITMAN·EXECUTIVE COORDINATION: KLÜVER

Bell Laboratories

Bell Labs transistor 1947

Intel i7 2011

2,270,000,000 transistors

Nude - Ken Knowlton and Leon Harmon - 1965

1
2

Lillian Schwartz

Pixillation - Lillian Schwartz - 1970

Pixillation - Lillian Schwartz - 1970

Lillian Schwartz - Olympiad - 1971

Lillian Schwartz - Night Scene - 1975

Lillian Schwartz

GIGLIX

TVC - Edward Zajec - 1971

TVC - Edward Zajec - 1971

il cubo: tema e variazioni

fig. 1

Tema e variazioni del cubo è un programma basato su un sistema di combinazioni operate con otto elementi base su una superficie reticolata. Gli otto elementi base sono stati derivati dalla scomposizione di un cubo in proiezione obliqua. I lati anteriore e posteriore del cubo sono paralleli al piano di proiezione, mentre i lati rimanenti sono in direzione obliqua (Fig. 1). A seconda della loro funzione questi elementi si possono classificare in tre categorie:

1. Elementi formativi (1)

2. Elementi connettivi (2, 3, 4, 8*)

3. Elementi additivi (5, 6, 7)

* L'elemento numero otto è derivato dalla rotazione dell'elemento numero tre.

TVC - Edward Zajec - 1971

Diagonal White 13685 - Edward Zajec - 1971

Edward Zajec

15. Prostor Variation 12
16. Prostor Variation 20

The program for both drawings was written in FORTRAN IV and executed on a Calcomp plotter controlled by an IBM 1130.

The program allows the following variations:

1. The basic shape of the composition can be any kind of rectangle (the square has been included as an exception) whose sides meet certain proportional relationships established by a random choice from a point in the Fibonacci series.

2. Such a rectangle (or square) is further divided into from a minimum of two to a maximum of eight partitions (the number of partitions in each composition is chosen at random), whose size and proportional relationships are also established at random from a point in the Fibonacci series.

3. A random choice is then effected between eight given units (or modules) which are consequently fitted to the dimensions of the respective partitions (scaling).

4. The grid lines between two partitions are drawn only if the lines between the two neighboring units do no coincide.

'DIAGONAL WHITE 75635'

MARIO
000300
×01
WORLD
1-1
TIME
379
SOFTPEDIA

Mario Soup - Ben Fry - 2003

1UP
Vol
100030500 ABCDEF
GHIJKLMN-PQRSTUV
WXYZ0123456789Km/h
DE
TV

1UP
Vol
100030500 ABCDEF
GHIJKLMN-PQRSTUV
WXYZ0123456789Km/h
1986DATESCORPHI.

1UP
Vol
100030500 ABCDEF
GHIJKLMN-PQRSTUV
WXYZ0123456789Km/h

1UP
Vol
100030500 ABCDEF
GHIJKLMN-PQRSTUV
WXYZ0123456789Km/h
NE
XT

```
void setup() {
  size(800, 600);
  stroke(255, 204);
  smooth();
  background(0);
}

void draw() {
  float thickness = dist(mouseX, mouseY, pmouseX, pmouseY);
  strokeWeight(thickness);
  line(mouseX, mouseY, pmouseX, pmouseY);
}
```

An Element is a si
one or more Behavi
Elements and deter
Elements are visua
of a circle and on
line at a constant
Element 1 and draw

Forms
F1: Circle
F2: Line

Behaviors
B1: Move in a straight line
B2: Constrain to surface
B3: Change direction while touching another Element
B4: Move away from an overlapping Element
B5: Enter from the opposite edge after moving off the surface
B6: Orient toward the direction of an Element that is touching
B7: Deviate from the current direction

Elements
E1: F1 + B1 + B2 + B3 + B4
E2: F1 + B1 + B5
E3: F2 + B1 + B3 + B5
E4: F1 + B1 + B2 + B3
E5: F2 + B1 + B5 + B6 + B7

Process 12 - Casey Reas - 1509

Process 12 - Casey Reas - 2006

Process 12 - Casey Reas - 2006

A rectangular surface filled with instances of Element 3, each with a different size and gray value. Draw a dot at the center of each line. Draw a quadrilateral connecting the endpoints of each pair of Elements that are touching. Increase the opacity of the dot and quadrilateral while the Elements are touching and decrease while they are not.

Casey Reas
Fall 2006