G-PASCAL NEWS

Published by Gambit Games for Commodore 64 G-Pascal owners.

Mail: G-Pascal News, P.O. Box 124, Ivanhoe, Victoria 3079. (Australia)
Phone: (03) 497 1283

Gambit Games is a trading name of Gammon & Gobbett Computer Services Proprietary Limited, a company incorporated in the State of Victoria.

VOLUME 1, NUMBER 1

Welcome to the first edition of the Commodore 64 G-Pascal News - the newsletter for owners of Commodore 64 G-Pascal! This is an official publication of Gambit Games for the support of G-Pascal - in it we hope to achieve a number of aims:

* To distribute information about how to do various things in G-Pascal (such as the bulletin board communication program in this issue).
* To answer readers' queries.
* To expand on material in the G-Pascal Manual that users feel is not clear or explained in sufficient detail.
* To print letters and comments from readers which may be of interest.
* To advise of other Gambit Games products.

We would be particularly interested to receive letters from users asking 'How do I ...' so that we can print the question and reply. Hopefully this will increase knowledge of the capabilities of G-Pascal to all users.

If you have a problem or a grumble please let us know also - it may be the result of a misunderstanding (perhaps a confusing explanation in the User Manual) - we would like to be able to clear them up if possible.

Accompanying this issue is a free index to the G-Pascal Manual.

Reprinted April 1984. Please address correspondence to **G-Pascal News, P.O. Box 124, Ivanhoe, Victoria 3079.**

OTHER G-PASCAL PROGRAMS

Gambit Games now supplies on the G-Pascal disk various programs written in G-Pascal which will be of interest to G-Pascal owners. Prior to serial number 7400 not all of the extra programs were supplied. In this case, an 'update disk' is available direct from Gambit Games for $20. Details and an order form should be enclosed with this copy of G-Pascal News.

ADVENTURE GAME

This is a 784-line adventure program written in G-Pascal. (A similar program for the Apple G-Pascal appeared in August 1982 'Your Computer' magazine). The game is supplied in source code format ready for you to load and compile.

The game is a complete adventure in its own right, but is really intended as a basis for your own ideas. The procedures necessary for taking and dropping objects, and moving from room to room are already there. All you need is to incorporate your ideas (room descriptions, new objects etc.) The program is very easy to follow - it makes extensive use of the CASE statement for decision-making.

SPRITE EDITOR

This program makes it easy to create and edit your own sprite shapes. It features a large grid on which the sprite shape appears, as well as the sprite in actual and double size. You can move a cursor around the grid with the keyboard or joystick and easily turn dots on or off. Sprite shapes can be saved to disk (or cassette) as a 1-sector file for later retrieval and re-use. Also, the program (on request) **will output DEFINESPRITE statements** for direct incorporation into your G-Pascal program, saving keying time and transcription errors.

The source code is supplied so you can examine and modify the program if you wish.

SOUND EFFECTS EDITOR

This program allows you to experiment with the synthesizer in your Commodore 64. It allows all three voices to have their attributes changed at the touch of a key, and played simultaneously or on their own. The program is 'menu driven' for extreme simplicity of use. It is not designed to play a tune, rather experiment with single sounds (such as an explosion, or gunshot

sound), for later incorporation into your programs and games.

The source code is supplied so you can examine and modify the program if you wish.

RUNTIME SYSTEM

The runtime system (which is mentioned in the User Manual) is ideal for those G-Pascal owners who would like to develop programs for sale, or for use in situations where having to recompile each time is a nuisance (such as educational software).

Use of the runtime system is straightforward - you just compile your program, save its P-codes to disk with the (O)bject option in the Files Menu, and then run the 'runtime create' program that is supplied with the runtime system. It asks for the name of your P-codes, attaches them to a copy of the P-code interpreter, and then saves the lot to disk or cassette with a short automatic loading routine.

There is provision for one program under the runtime system to load another and transfer control to it when the first program ends, thus effectively 'chaining' from one program to the other. This would be very handy for larger, menu-driven systems.

The runtime system includes permission to distribute the combined runtime system and P-codes under specified circumstances.

HINTS

LOADING/SAVING FILES

To avoid errors when loading or saving files from within your program, make sure that **all sprites are inactive.** This is a hardware limitation - active sprites cause additional memory accesses, thus slowing down the processor, possibly allowing it to get out of synchronization with the disk drive or cassette player.

INSERTING BLANK LINES

To insert blank lines in your G-Pascal program just type SHIFT/SPACE (i.e. space while pressing the shift key) and then press RETURN.

CHANGING EDITOR COLOURS

If you don't like the blue background to the Editor, patch byte $9B78 to a number from 0 to 15 which represents the colour you desire. If you don't like the white letters that the Editor uses, patch byte $8CD9 to the following: 5 - white, 28 - red, 30 - green, 31 - blue. For example, the following program will change the default system colours to green letters on a black background:

```
begin memc [$9B78] := 0;
      memc [$8CD9] := 30 end.
```

(This patch only applies to G-Pascal Versions 3.0 and 3.1).

JOYSTICK PROBLEM

Users of Version 3.0 G-Pascal may experience unexpected 'Break ...' messages in programs that access the joystick frequently (or the paddle fire buttons) by using the JOYSTICK function. This may occur if the keys 'X', 'V', 'N', comma or CRSR up/down are pressed. To correct this problem the following statement should be executed (once) by the program before referring to the joystick:

```
memc [$A80B] := $7F;
```

Note: this applies only to Version 3.0 G-Pascal users.

SAVING PROGRAMS TO DISK

Contrary to what earlier versions of the user Manual may state on pages 40 and 41 ('File Handling' chapter) saving a program to disk with the same name as one already there will **not** replace the existing program, instead a 'file exists' error will occur. To replace an existing program we suggest you delete the existing file first, then save the new one. i.e. enter 'D' (for DOS), then: s0:filename (to scratch the file). It is sound practice to keep more than one copy of your programs, in case a disk or cassette error occurs when reading that program back in later. We suggest numbering successive versions of your programs, so you might save an adventure game as: ADVENTURE 1, ADVENTURE 2, ADVENTURE 3 and so on. Later, if the disk is filling up, you can delete the older (lowest numbered) versions.

MEMORY ALLOCATION WITH IEEE

G-Pascal owners who are using IEEE disk drives may experience memory allocation problems. This is because G-Pascal moves $2E to memory location 1 (6510 input/output register) which disables Basic, but enables the Kernal ROM. If you wish

the Kernal ROM to be disabled (because special disk software resides in RAM instead), then you must patch location $9BC5 to $2C, by entering and running the following program before accessing the disks:

```
begin memc [$9BC5] := $2C end.
```

You should **only** enter this program if you believe you meet the above conditions. This patch is for Version 3.0 and 3.1 - please contact Gambit Games if you have a different version and you have this problem.

SQUARE ROOTS CALCULATIONS

It is sometimes handy in programs to be able to calculate the square root of a number, for example the distance between two points (the hypotenuse of the triangle) is calculated by taking the square root of the sum of the squares of the distances between them in the X and Y directions. The program on the right calculates integer square roots. It illustrates how you can improve the accuracy of calculations involving integer arithmetic by appropriate 'scaling'. In other words, the square root of 2 is the same as the square root of 200 divided by 10.

TERMINAL COMMUNICATION PROGRAM

The program over the page is designed to allow your Commodore 64 to act as a 'full duplex' terminal, when connected to another computer via a modem.

For correct operation an RS232 interface should be plugged into the user port of the Commodore (these are available from Commodore dealers for around $45). Then a suitable cable is connected between the RS232 interface and a modem, which is connected to the phone line.

The program would be suitable for communicating with the 'Source', the 'Beginning', MICOM's CBBS (Computerised Bulletin Board Service) and so on. To convert the program to half-duplex operation (so that what you type is echoed on the screen immediately), just add the statement:

```
WRITE (CHR(X));
```

between lines 70 and 71.

```
(* Program to demonstrate how to
   obtain square roots in
   G-Pascal. to improve accuracy,
   output from sqrt is 128 times
   the actual result. for the
   correct result, divide by 128
   (or shift right 6 times).

*)

VAR j, k : INTEGER ;

FUNCTION sqrt ( x );
(*----------------*)
VAR sq, a, b, i : INTEGER ;
BEGIN
IF x = 0 THEN
  sqrt := 0
ELSE
  BEGIN
  x := x SHL 12;
  sq := ABS (x);
  a := x;
  b := 0;
  i := 0;

  WHILE a <> b DO
    BEGIN
    b := sq / a;
    a := (a + b) SHR 1; (* divide by 2 *)
    i := i + 1;
    IF i > 4 THEN
      BEGIN
      i := 0;
      IF ABS (a - b) < 2 THEN
        a := b
      END
    END ;
  sqrt := a
END
END ;  (* sqrt *)

(*------------------------------*)
(* Program starts here - display
   square roots of first 200
   numbers to 2 decimal places.
*)

BEGIN
  FOR k := 0 TO 200 DO
    BEGIN
    j :=(sqrt (k) * 100) SHR 6;
    WRITE (" sqrt(", k, ") = ",
           j / 100, ".");
    IF j MOD 100 < 10 THEN
      WRITE ("0");
    WRITELN (j MOD 100)
    END
END .
```

```
(* program to access bulletin boards *)

(* Author: Nick Gammon *)

CONST
      cr = 13;
      lf = 10;
      home = 147;
      true = 1;
      false = 0;
      areg = $2b2;
      xreg = $2b3;
      yreg = $2b4;
      setlfs = $ffba;
      setnam = $ffbd;
      openit = $ffc0;

VAR name : ARRAY [1] OF CHAR ;
    input : CHAR ;

PROCEDURE open_rs232_file;
(************************)
BEGIN
  MEMC [$f8] := $c1; (* buffer *)
  MEMC [$fa] := $c2; (* buffer *)
  MEMC [areg] := 2;
  MEMC [xreg] := 2; (* RS232 *)
  MEMC [yreg] := 2;
  CALL (setlfs);
  MEMC [areg] := 2;
  MEMC [xreg] := ADDRESS (name[1]);
  MEMC [yreg] := ADDRESS (name[1]) SHR 8;
  CALL (setnam);
  CALL (openit)
END ;

FUNCTION from_modem;
(******************)
BEGIN
  GET (2);
  from_modem := GETKEY ;
  GET (0)
END ;

PROCEDURE display_char (x);
(************************)
BEGIN

(* Reverse upper/lower case *)

  IF (x >= $61) AND
     (x <= $7a) THEN
     x := x - $20
  ELSE
  IF (x >= "a") AND
     (x <= "z") THEN
     x := x + $20;

(* Only display if printable *)

  IF ((x >= " ")
  AND (x <= $7f))
  OR (x = cr)
  OR (x = lf) THEN
     WRITE (CHR (x))
END ;

PROCEDURE to_modem (x);
(*********************)
BEGIN
  PUT (2);
  WRITE (CHR (x));
  PUT (0)
END ;


(* -------- MAIN PROGRAM -------- *)
BEGIN

(* first set up the file name
   as per the RS232 paramters *)

  name [1] := 6;  (*  300 baud  *)
  name [0] := 0;  (*  3-line    *)
  open_rs232_file;

(* now that the file is open ,
   loop reading characters from
   the modem and displaying them
   on the screen. At the same time
   get characters from the
   keyboard and send them to the
   modem.
*)

  WRITE (CHR (home));
  REPEAT

(* process input from the modem.
   (0 means none)
*)

    input := from_modem;
    IF input <> 0 THEN
      display_char (input);

(* process input from the keyboard
   (0 means none).
*)

     input := GETKEY ;
     IF input <> 0 THEN
       to_modem (input)
  UNTIL false

END .
```