Mike O' Connell
5835-A Brett
Fort Knox KY 40121

Tiger Trouble!

TI programmable-calculator safari

Your long-awaited camera safari has been a great success. Now, though, your party faces potential tragedy. While traveling down the Ngabiki River, one of the boat's crewmen has fallen seriously ill, and your guide fears he may die without medical attention. Fortunately, a doctor lives in a nearby village, which can be reached from the next landing dock along the river; *un*fortunately, a legendary man-eating tiger is rumored to roam the region between the dock and village.

Someone must undertake the dangerous journey to the village and return with the doctor. You have — naturally — volunteered. Unarmed, you hope to avoid the tiger — if it exists. The other members of your party salute your courage and leave you at the dock with wishes of luck and speed. As you stand trying to choose between the two paths leading away from the dock toward the village, you hear the tiger roar.

Tiger Trouble reverses the traditional game roles, in which the player normally hunts or tracks an elusive prey, through the combined use of random number generation and conditional transfer functions. For more information on the Texas Instruments programmable calculators, their conditional transfer functions and random number generation, see Pete Stark's articles in issues 1 and 2 of *Kilobaud* (Jan. and Feb. 1977). Briefly, the calculator employs a simple algorithm to generate pseudo-random numbers.

In this program, as in Pete Stark's Submarine game (*Kilobaud*, No. 2, p. 70), a seed number between 0 and 1 is multiplied by 7^9 (40,353,607), and then divided by 10^5 (100,000).

Digits to the left of the decimal point are discarded, and the remainder is a random number between 0 and 1 (and also the new seed number).

Multiplying the first random number by 21 and rounding to an integer gives an initial location for our tiger; successive random numbers between 0 and 1 are subsequent moves by the tiger; sucessive random numbers between 0 and 1 are multiplied by 4 and then rounded to an integer value of 0, 1, 2 or 3. Here is where the conditional transfer function operates to reverse traditional game roles.

If the tiger's location is numerically greater than the player's location, the integer value generated (0,1,2,3) is subtracted from the tiger's location; otherwise, it is added to the tiger's location. In other words, the tiger will move 1, 2, or 3 spaces closer to the player's position, or will remain in place after each move the player makes. Other conditional transfers are used to end the game if the tiger catches the player, and to prevent the tiger from straying into the Ngabiki River (negative numbers). See Fig. 1.

Game Rules

The basic operating rule of the game is that the tiger stalks the player. Rules of play and comments follow.

1. A player may take any route to reach the village and return (see the game board configuration, Fig. 2), provided he moves one numbered location at a time along a solid path. A player need not move to locations in numerical sequence and may go forward or backward.

2. In addition to the landing and the village, the game board contains four sectors: the swamp (locations 1-5), forest (locations 6-10), plains (locations 11-15) and hills

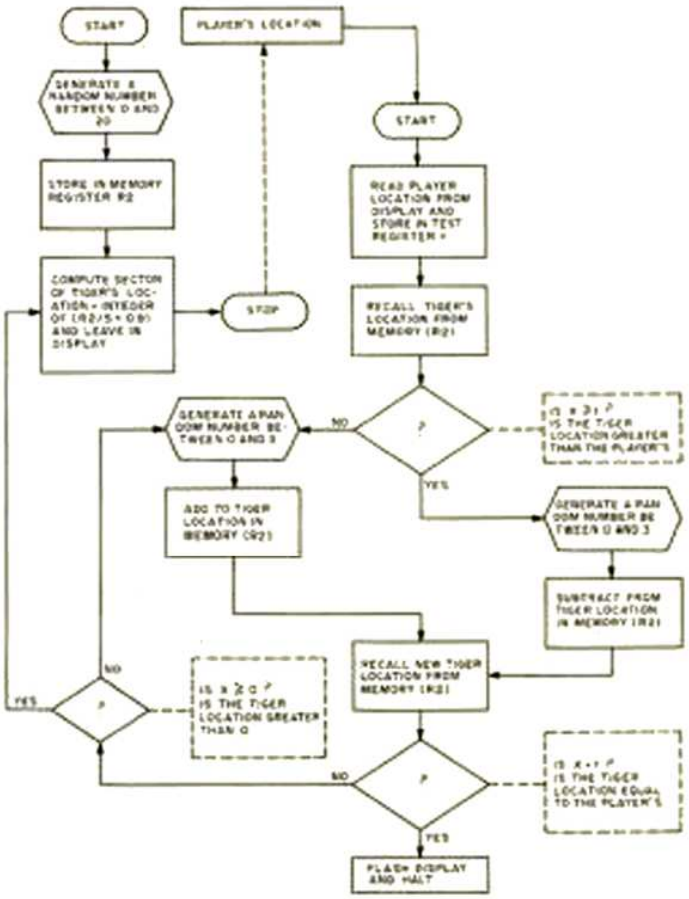

Fig. 1. Flowchart.

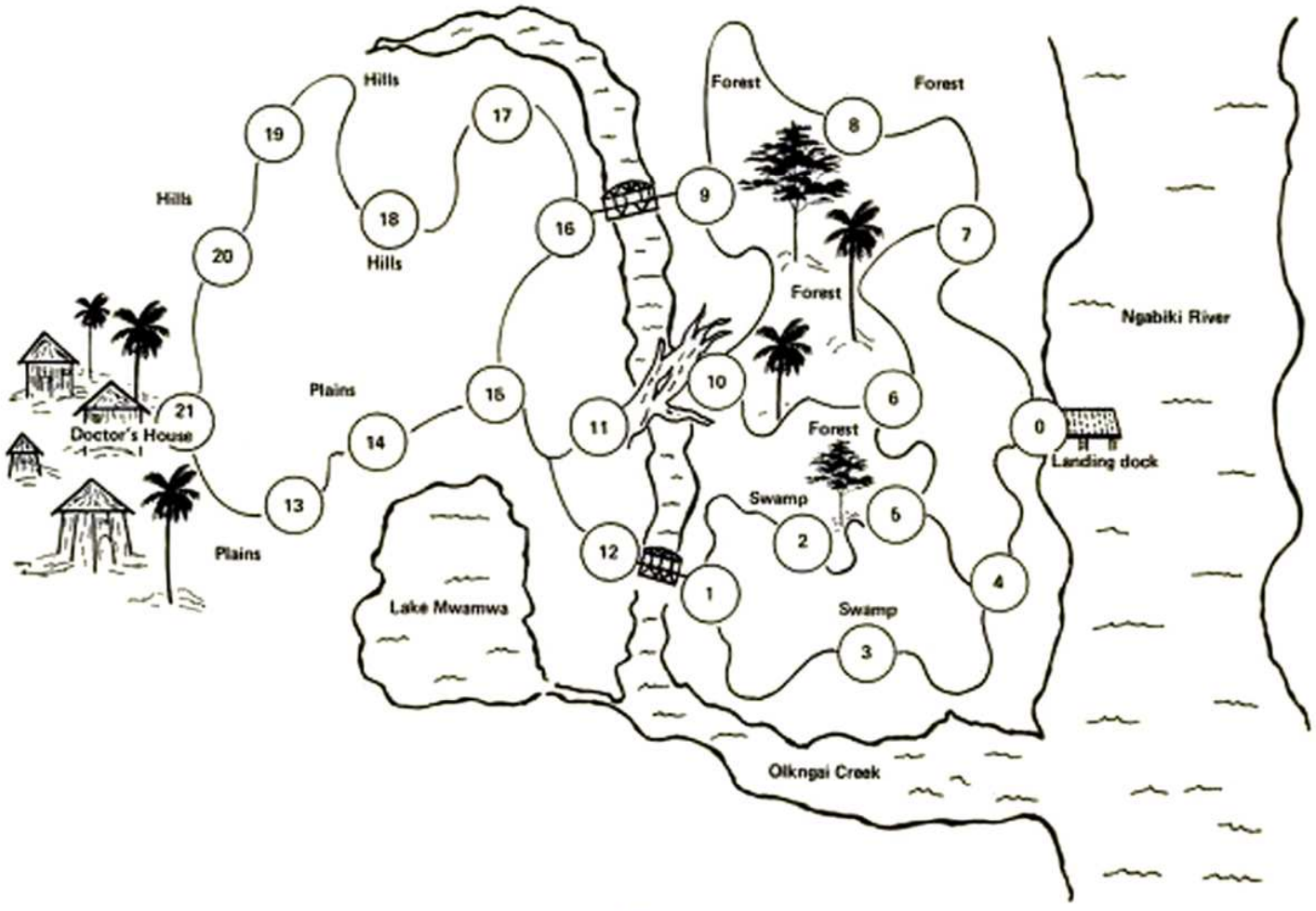

Fig. 2. Game board.

Location	Instruction	Comments
00	21 Sto 8	Set upper limit of random number at 20.
04	Subr 65	Get a random number between 0 and 20
07	Sto 2	. . . and store in memory register 2 as the tiger's initial location.
09	4 Sto 8	Reset upper limit of random number at 3.
12	Subr 44	Compute and hold for display the integer representing the sector of the tiger's location, and
15	R/S	Halt. Wait for player to enter first location.
16	x ≷ t	Take player's new location from display and place in test register t.
17	Rcl 2,x ≥ t	Recall tiger's location and compare with player's location in test register,
20	54	. . . if tiger's location is greater, transfer to instruction 54.
22	Subr 65	. . . if tiger's location is less, get a random number between 0 and 3,
25	Sum 2	and add to register 2 to adjust the tiger's location.
27	Rcl 2, x = t	Recall the tiger's new location and compare with player's location in test register,
30	63	. . . if the two locations are the same, transfer to instruction 63 to flash the display.
32	CP	. . . if the two locations are not the same, set the test register to 0,
33	Inv x ≥ t	and check to determine if the tiger's location is less than 0,
35	22	. . . if the tiger's location *is* less than 0, return to instruction 22 (to get a random number between 0 and 3, add to tiger's location, and repeat checks for same locations and a negative tiger location).
37	Subr 44	. . . if the tiger's location is not less than 0, transfer to instruction 44 for the subroutine to compute and hold for display the integer representing the sector of the tiger's location, and
40	R/S	Halt. Wait for player to enter his next location.
41	Gto 16	When new player location has been entered, transfer to instruction 16 (to execute sequence 16 to 40).
44	Rcl 2 ÷ 5 + .9 =	Start of subroutine to calculate the integer representing the sector of the tiger's location, which is equal to
52	Int Rtn	the integer of the value of $(R_{2/5} + 0.9)$.
54	Subr 65	Get a random number between 0 and 3,
57	Inv Sum 2	and subtract it from the value of the tiger's location in memory register 2, and after adjusting tiger's
60	Gto 27	location, transfer to instruction 27.
63	0 1/x	Flashes the display (1/0 does not calculate).
65	Rcl 9 X 7 y^x 9 X	Start of subroutine to generate a ran-
72	5 ± 10^x =	dom number . . . calculates (seed X 7^9)/10^5,
76	Inv Int Sto 9	and discards digits to the left of the decimal point, to get a random number between 0 and 1, which is stored in register 9 as the new seed,
80	X Rcl 8 =	and multiplied by the value in register 8, which is equal to one more than the value of the desired upper limit of the random number, and that product equals a random number between 0 and the number in register 8, which is
84	Int Rtn	converted to an integer within the desired range, and returned to the main program routine.

Fig. 3. Program listing.

Player's Move	Display	Comments
- -	1	Tiger is initially in the swamp — take the forest path.
7	1	Tiger is still in the swamp — continue toward

(locations 16-20). After each set of moves by player and tiger, the tiger will roar, partially betraying its location. This roar is "heard" as a displayed integer, indicating the sector in which the tiger is located. (A display of 1 means the tiger is in the swamp; 2 means forest; 3 means plains; 4 means hills. The tiger may go to the landing dock, and the display will be 0, or enter the village, causing a 5 to be shown.)

3. Both player and tiger cross Olkngai Creek. The player crosses at the bridge connecting locations 1 and 12, or at the one between locations 9 and 16; because these are man-made and appear strange, the tiger will not cross them. The tiger will cross only at the fallen tree between locations 10 and 11. The player cannot cross this.

4. Unlike the player, the tiger does not travel on the paths; instead, it travels freely through the area to arrive at numbered locations on a path. And only when player and tiger arrive at the same location *at the conclusion of their moves* does the tiger bring an abrupt end to the game (and player). Example: If the player moves to location 9 and the tiger then moves from location 7 to location 10, it will *not* have encountered the player en route and ended the game.

5. The plains is probably the most dangerous sector of the board. Although the path is shorter than the one through the hills for player and tiger alike, the numbering system gives the tiger a distinct mobility advantage in this sector (not unrealistically, since the tiger can best capitalize on its greater speed in wide, open expanses).

6. The game ends when the player returns to the landing dock after having first gone to location 21 to get the doctor. Or, if the display flashes 9.999999999 99 continuously after a move, the tiger has arrived at the same location as the player, and,

well . . . sorry, but the game ends that way, too.

It appears simple, but the odds (and the program) are with the tiger, and it is difficult to win. To play, note the rules and comments above, enter the program as shown in Fig. 3 (remember to put a seed number in memory register 9) and press the R/S button. You are now at the dock, and the number displayed represents the sector of the tiger's initial location. To move, enter the number of the location to which you wish to advance and press R/S. That's all there is to it.

Good luck and safe journey! ■

		the bridge at location 9.
8	2	Tiger is pursuing and is in the forest.
9	2	At the bridge — tiger also still in forest.
16	2	Tiger in forest — take mountain (hills) path.
17	2	Tiger still in forest — must be lost — continue.
18	3	Tiger pursuing — crossed log into plains area.
19	3	Tiger searching plains — continue on hills path.
20	4	Tiger has entered hills in pursuit.
21	4	Reached doctor — must return to dock — since tiger in hills, take plains path.
13	3	Oops — tiger returned to plains — back to village to avoid.
21	4	Tiger has returned to hills — has us well boxed in — will risk the plains path.
13	3	Tiger is back in plains — proceed with caution.
14	3	Tiger is searching plains — will risk continuing.
15	3	Tiger still searching plains — head for bridge at location 12.
12	3	At the bridge — tiger still searching plains — cross into swamp.
1	3	Across bridge — tiger still on plains — should be safe — continue to dock.
2	2	Oh-oh — tiger has crossed log into forest in pursuit — head for dock.
5	2	Tiger moving through forest — race for dock.
4	2	Tiger still in forest — just one more move to landing dock.
0	2	Reached dock safely — tiger never made it through the forest — score one for us.

Fig. 4. Sample run (with seed = 0.9). Note: Entire sequence should be duplicated with the same initial seed and identical moves, and can be used as a program check.