MASSACHUSETTS INSTITUTE OF TECHNOLOGY
Electronic Systems Laboratory
Department of Electrical Engineering
Cambridge 39, Massachusetts

MEMORANDUM 8436-M-29

TO: D.T. Ross

FROM: H.R. Morse

SUBJECT: Preliminary Operating Notes for SAGA II

DATE: October 19, 1960

This memorandum presents, in condensed form, the features of the control and operating program for SAGA II, the TV script-writing program. The control routine permits modification, printing and punching of "switch" values; tracing the operation of SAGA at several levels of detail; and examination and modification of certain desired quantities, all under control of the on-line Flexowriter. No program details are given in this brief memorandum. A complete description of the design and programming of SAGA II will be issued later as a Technical Memorandum.

Starting Procedure:

Upon read-in, control is automatically transferred to the control routine.

Typing the character r followed by a carriage return (↵) will reset and start SAGA.

The run number typed is the contents of the random number generator at the start of the play, and may be examined and modified by typing r=. Starting SAGA with the same switch settings, and the same number in the random number generator will reproduce a play.

The timing number initially controls the length of time until the sheriff arrives. If it is positive the sheriff will arrive immediately after the robber; the more negative the timing number, the longer until the sheriff arrives. This timing number may be examined and modified by typing t=, and is initially set to -1000.

Switches

A "switch" is a probability branching device which controls the sequencing of action in SAGA. The state of the action is represented by 16 "state variables" which are multiplied by weights (a-values) to determine the probability of taking any given branch of a switch. Considering switch n, the value for branch k is

$$P_{n,k} = a_{n,k,0} + \sum_{j=1}^{16} (a_{n,k,j}) \cdot (b_j)$$

where the b's are the state variables

The P's are scaled, by a simple procedure, to be fractions representing the probabilities for the respective branches.

Probabilities are calculated each time a switch is entered, since the state variable can change at any time, after which a random number is generated, and used to choose a branch. A list of state variables is given in Appendix 1.

A flow chart which explains the use of each switch can be found in the TX-0 computer room.

The "character" of the program is controlled by the switches, so by changing the a-values of a given switch, one may make the robber an excellent shot, or a lousy one; make the sheriff a fast actor and the robber a slow one; make the robber a good shot when drunk; or make the sheriff a brave man; to give a few examples. Examples of changing switches will be given later.

Switch Examination and Modification

sN↵ selects switch N.

sNr will reset switch N, and ask for the number of branches. Switch N may then be defined or redefined.

sNp selects switch N and prints the a-values for each branch, with the exception of most zero a-values.

After a switch has been selected, an a-value may be examined by typing aI,J=, where I is the branch number, and J the number of the associated state variable. The a-value may then be modified by typing the new value followed by a carriage return (↵), or left as is by just typing a carriage return. After modifying an a-value in a branch, other a-values in the same branch may be referred to by typing aJ= where J refers to the associated state variable. Switch numbers, branch numbers, and state variable numbers are in decimal, while the a-values are taken as octal, and should be typed as such. The value of A_o for any branch may be full range

$$-2^{17} \quad A_o \quad +2^{17},$$

while all other a-values are taken modulo 2^5.

Error Indications Possible When Modifying A-values

nbr	There exists no such branch
ado	The switch cannot accomodate another a-value at present. Redefining switch N by sNr will eliminate the problem
nrm	Means the a-table is full. Uncorrectable.

If a modified a-table produces good plays, it may be punched out by typing ap, and read into SAGA when desired. Switch 42 has been reserved for identification purposes; $a_{1,0}$ containing the octal date, and $a_{2,0}$ containing the number of the a-table punched on that date (first, second, etc.).

An Example of Modifying A Switch

```
SWITCH 26
    br1    a0 = 10      a12 = 30
    br2    a0 = 60      a5  = 1
    br3    a0 = 110     a1  = -10     a5 = 3
```

Switch 26 controls whether the sheriff hits (branch 1), nicks (branch 2) or misses (branch 3) the robber when he shoots.

State variable 1 is +10 if the sheriff sees the robber
-10 if he does not

" " 5 is 0 if the sheriff has not been nicked
+10 if he has

" " 12 is +10 if sheriff is using his last bullet
0 if he is not.

So if the sheriff is not using his last bullet, sees the robber, and has not been nicked, then the probabilities are

```
p1 = 10      (hit)
p2 = 60      (nick)          (1)
p3 = 10      (miss)
```

i.e. he is 6 times more likely to nick the robber then miss or hit him.

The sheriff could be made a good shot by setting $a_{1,0} = 60$ and $a_{2,0} = 10$. Then the probabilities for the above case become

```
p1 = 60      (hit)
p2 = 10      (nick)
p3 = 10      (miss)
```

Or he could be made a lousy shot by setting $a_{3,0} = 60$ and $a_{2,0} = 10$. Then we have

p1 = 10 (hit)
p2 = 10 (nick)
p3 = 60 (miss)

Also, it can easily be seen how changes in the state of the program would affect the probabilities on the branches, i.e. - if the sheriff were nicked (b5 = 10) then (1) would become

p1 = 10 (hit)
p2 = 70 (nick)
p3 = 40 (miss)

The a-values may be minus, but if a final branch-probability is minus, then the probability is set to zero. Notice also that there is no state variable zero. A_o is used purely to balance the branch, and is treated as if state variable zero were always 1.

Program Interruptions and Tracing

The remaining part of the control program will be mentioned only briefly here, as it was designed mainly for debugging purposes.

The program may be interrupted at any time by changing the position of TAC_o, which should be 0 upon read-in. SAGA will break at the next switch, and transfer to the control program. Any of the control operations may be performed, and the play continued by typing a period (.).

Tracing features:

Typing p will cause each switch number to be printed as it is used, followed by the probabilities calculated for each branch, and the play continued.

n will cause just the switch numbers to be printed, and the play continued.

b will cause the program to wait for a command from the flexo at each switch, and may be used in conjunction with p and n.

e will cause p,b,n to be erased.

. will always continue the interrupted play after a break.

sNb may be preceeded by p or n, and followed by b, all having the same effect as above when switch N is used. As many as 8 switch break points may be active at once.

sb erases all switch break points.

STATE VARIABLES

No.	Condition	Value	
		Yes	No
1	shf sees rob	+10	-10
2	rob sees shf	+10	-10
3	sheriff is coming	+10	0
4	rob is aiming	+10	0
5	sheriff nicked	+10	0
6	robber nicked	+10	0
7	sheriff hit	+10	0
8	robber hit	+10	0
9	inebriation factor	starts at zero, stepped by +10 for every drink from glass, +20 for every drink from bottle that rob takes.	
10	sequence	Stepped in varying units of +10 if rob acting, by -10 if shf acting.	
11	robber's last bullet	+10	0
12	sheriff's last bullet	+10	0
13	door open	+10	0
14	sheriff inside	+10	0
15	robber inside	+10	0
16	drink full	+10	0

SUMMARY OF COMMANDS

Typed Command	Effect
$r_{\not\Delta}$	reset and restart SAGA
r=	examine contents of random number generator
t=	examine initial setting of timing number
$sN_{\not\Delta}$	select switch N
sNr	reset switch N
sNp	select and print switch N
sp	print all switches
aI,J=	examine aJ for branch I of switch N
aJ=	examine aJ for last branch examined of switch N
ap	punch entire a-table
b	break at each switch entered
.	continue after break
p	print switch numbers and branch probabilities as used and continue play (unless command followed by b)
n	as p, except print just switch numbers
sNb	break only at switch N. (this may be preceeded by p or n, or followed by b, with the expected results). As many as 8 breaks may be used at once.
sb	erase all switch break points
e	erase all of the break commands
TAC_o	changing TAC_o will cause SAGA to break at the next switch used. This is a "one shot" break. As usual (.) will continue the play.