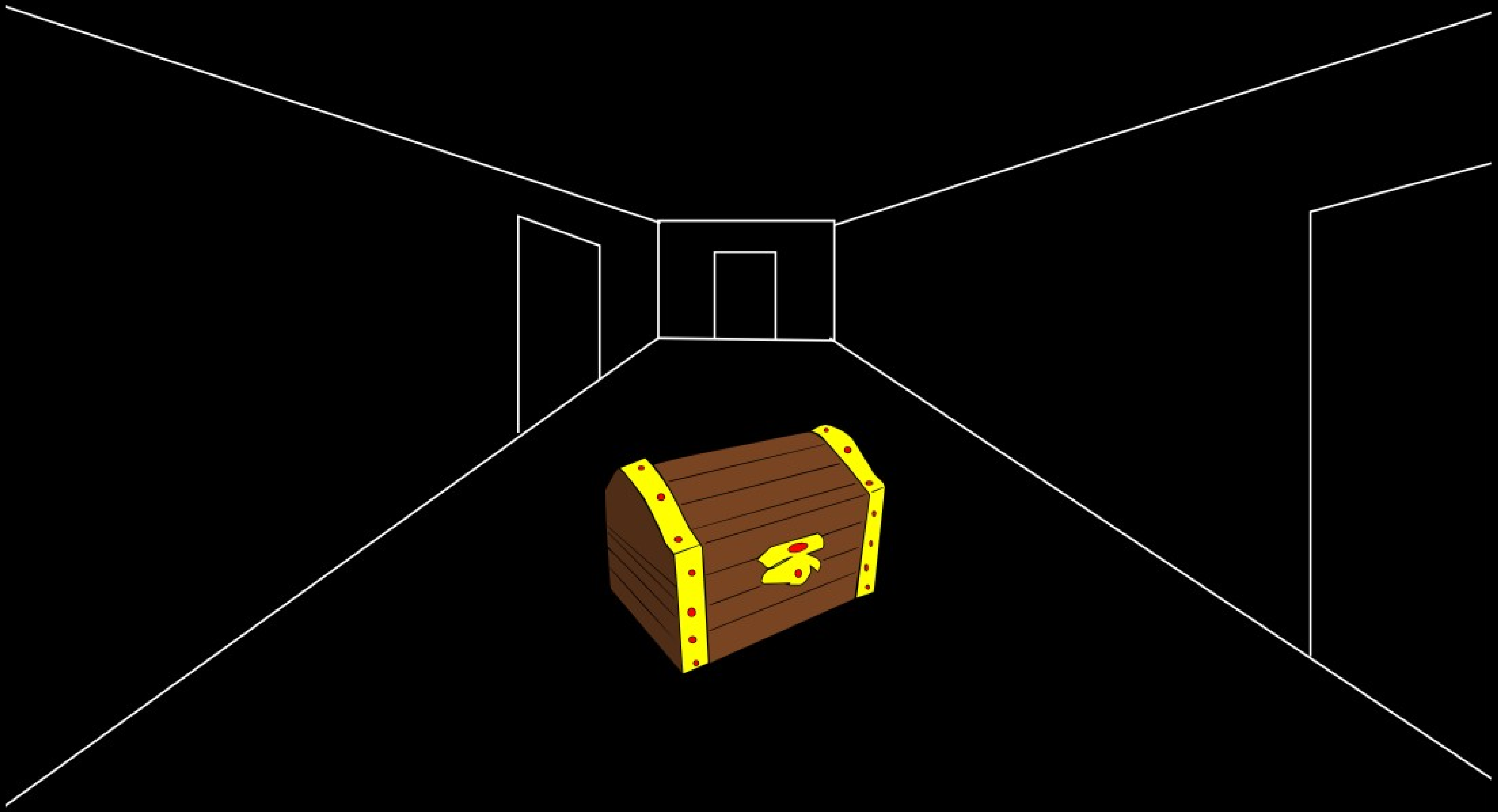
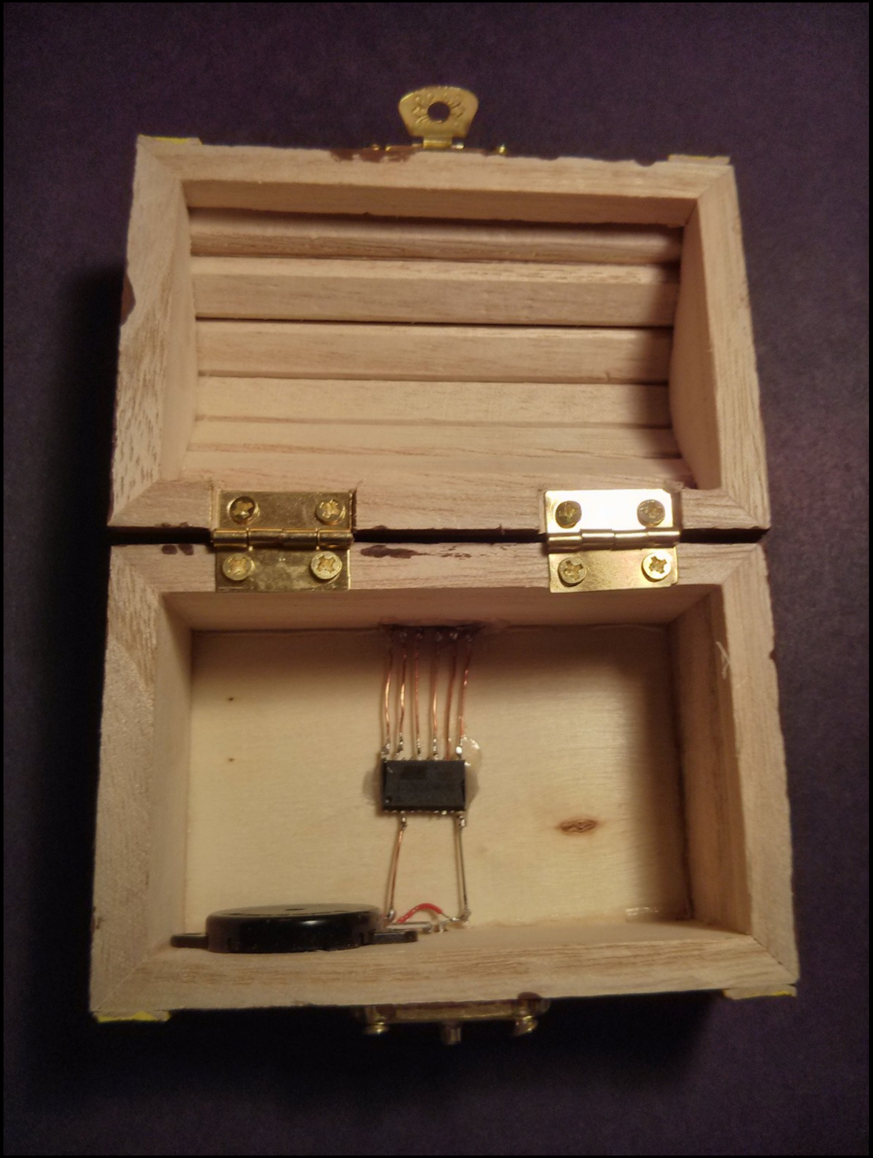
A Tiny Room in a Tiny World

Lee Djavaherian, speaker

2015 US IRDC, Atlanta, May 30-31





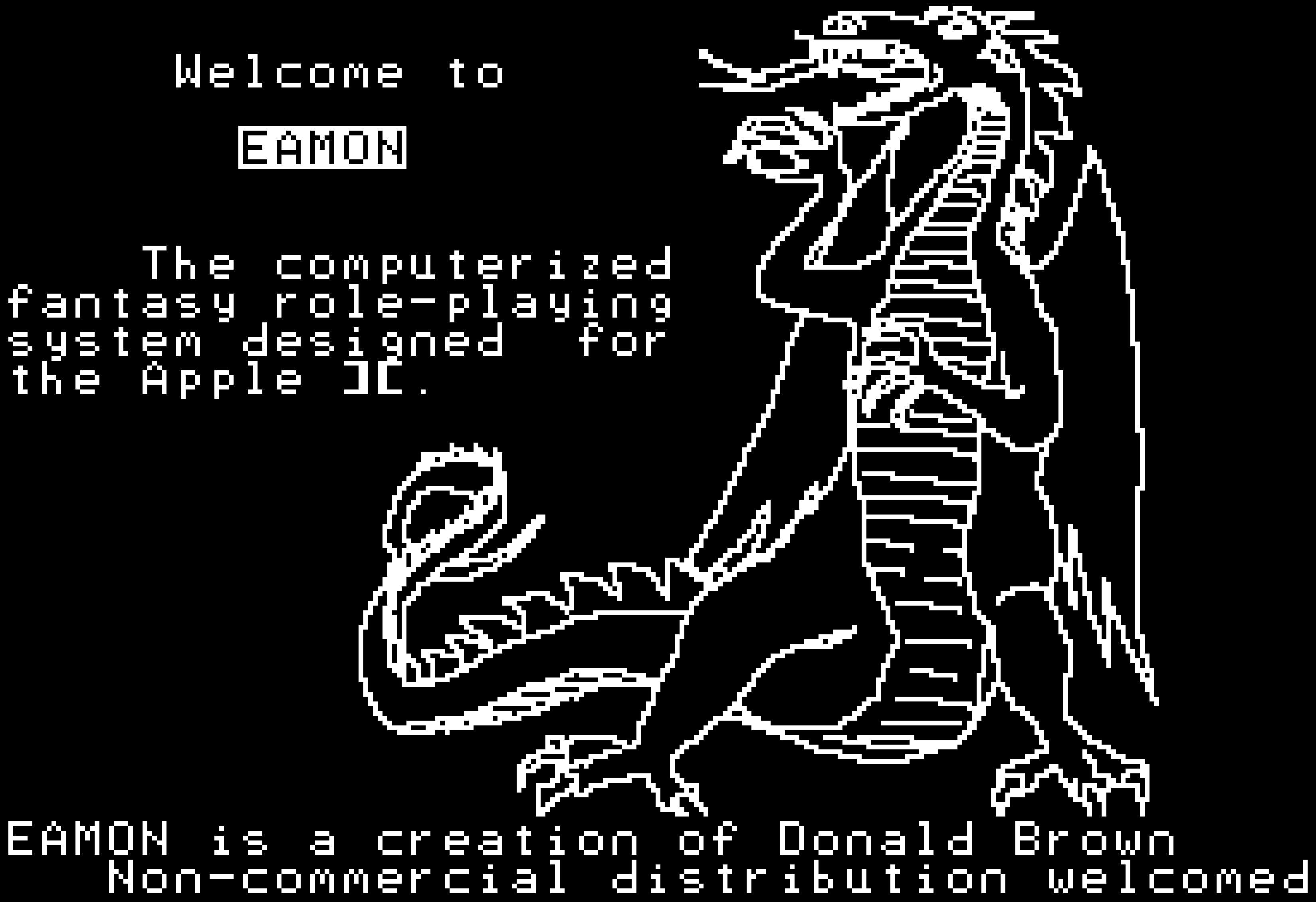
But if you are INSIDE the box, computation looks very different, it looks like movement, pattern, our lives.

Every secret in every box is just a new box with a new secret. There is no actual treasure here, just a 0 or a 1, a this or a that. The treasure is the pattern itself, the structure, the journey.

The Roguelike is a journey of computation, an abstraction of a dualistic universe; the adventurer and the place to adventure; the halls, the rooms, a giant fractal tree of information expression.

Eamon - The Beginners Cave (1980)

by Donald Brown



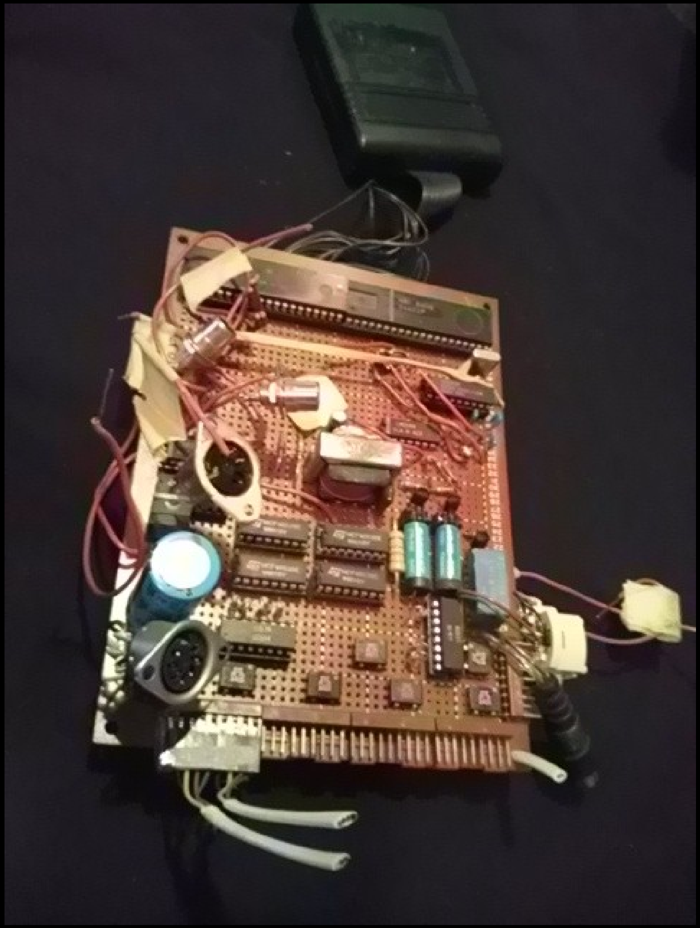
```
        Welcome to

          EAMON

     The computerized
fantasy role-playing
system designed  for
the Apple ][.

EAMON is a creation of Donald Brown
   Non-commercial distribution welcomed
```

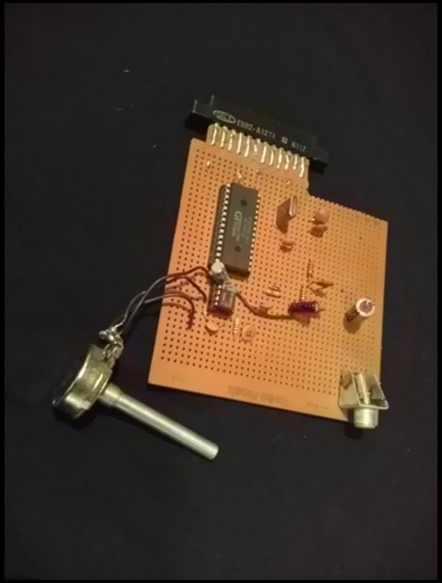
Golden age of electronic RPGs (1980-1982)

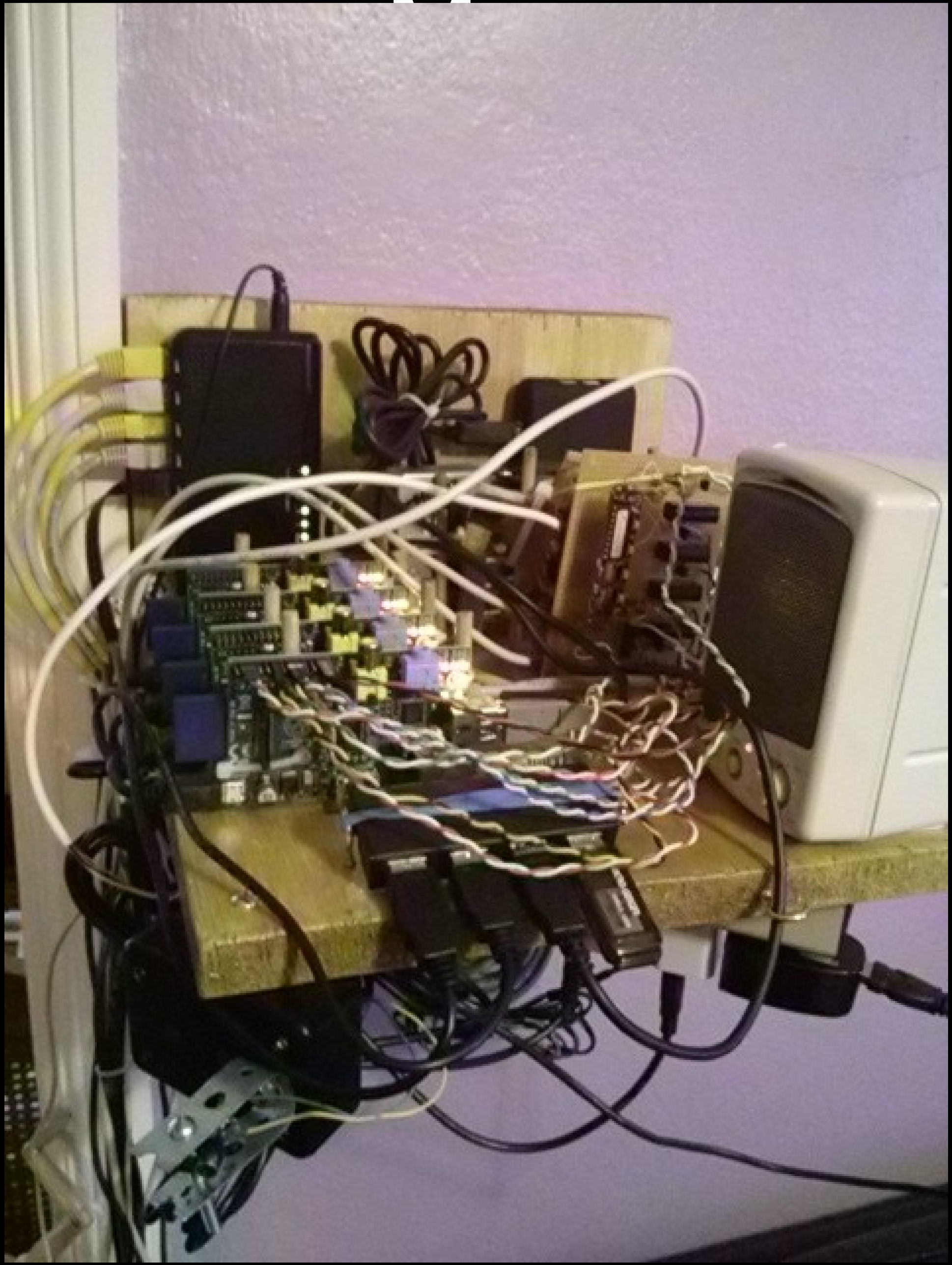
- Rogue, Unix, C, 1980
- Eamon, Apple II, BASIC, 1980
- Akalabeth, Apple II, BASIC, 1980
- Ultima, Apple II, BASIC & Assembly, 1981
- Dark Tower, electronic board game, Assembly?, 1981
- Tunnels of Doom, TI-99/4A, TI GPL, 1982
- Chase-N-Counter (Treasure Trek), electronic LCD game, Assembly?, 1982

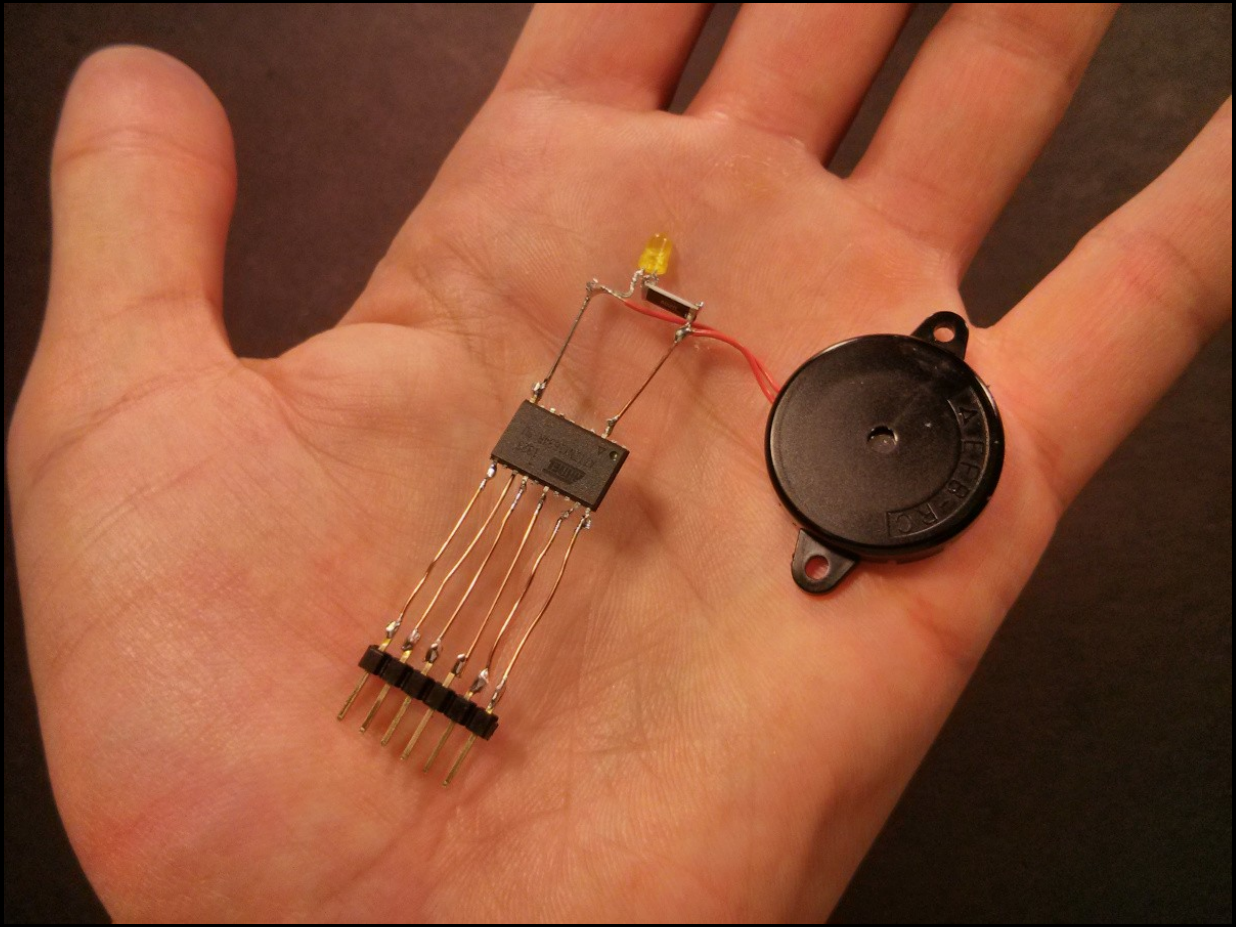
And then...

Video Game Crash of 1983



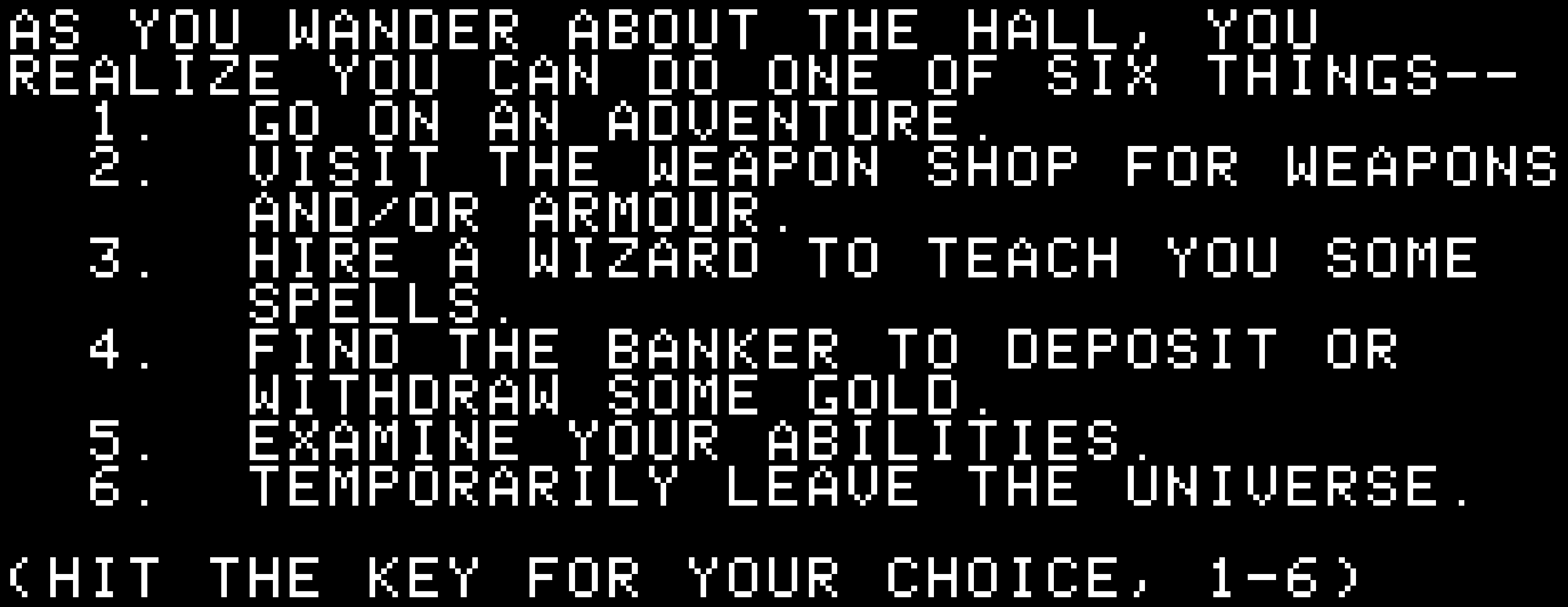






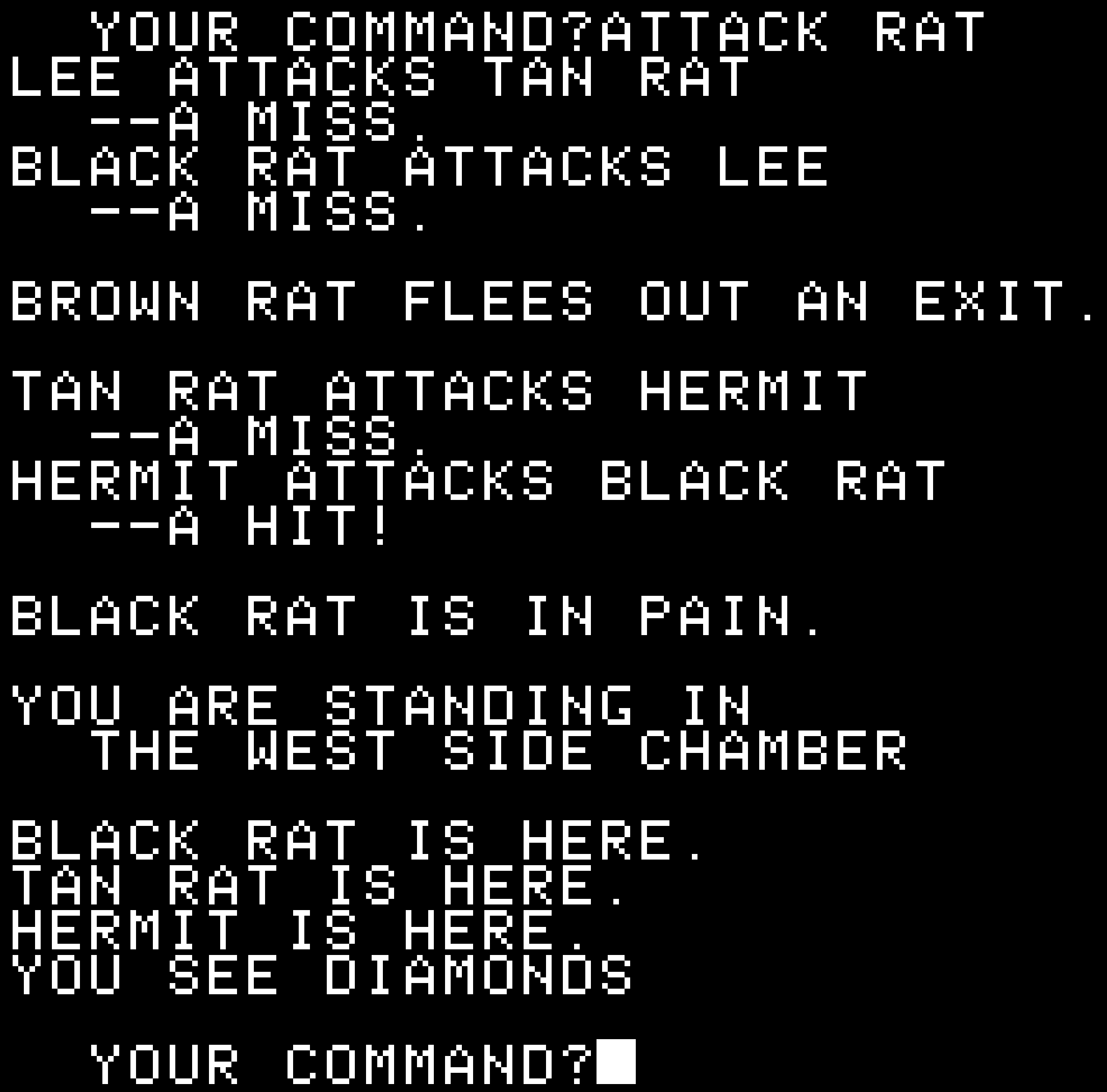
Eamon "town"

Hall of the Guild of Free Adventurers



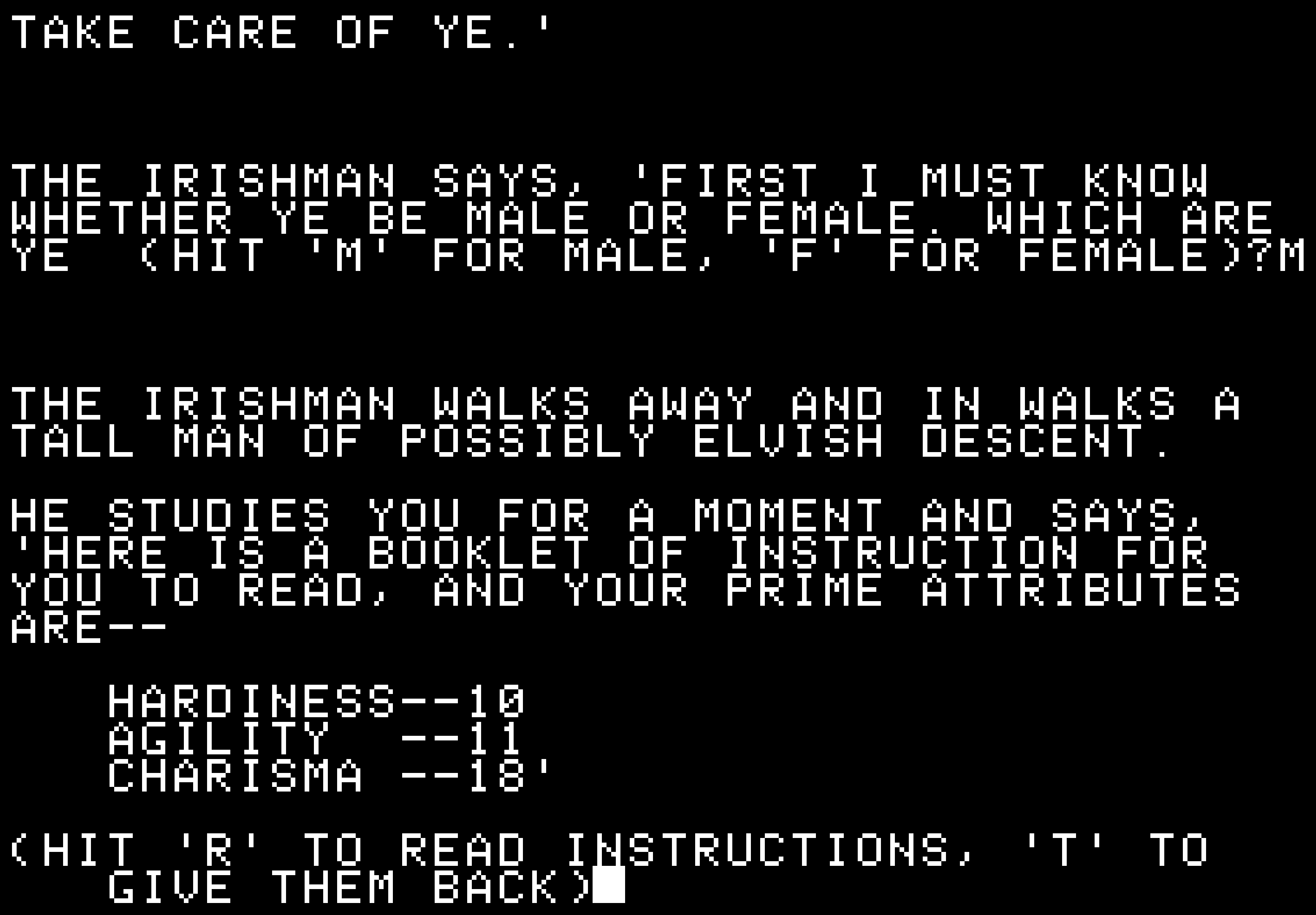
```
AS YOU WANDER ABOUT THE HALL, YOU
REALIZE YOU CAN DO ONE OF SIX THINGS--
  1.  GO ON AN ADVENTURE.
  2.  VISIT THE WEAPON SHOP FOR WEAPONS
      AND/OR ARMOUR.
  3.  HIRE A WIZARD TO TEACH YOU SOME
      SPELLS.
  4.  FIND THE BANKER TO DEPOSIT OR
      WITHDRAW SOME GOLD.
  5.  EXAMINE YOUR ABILITIES.
  6.  TEMPORARILY LEAVE THE UNIVERSE.

(HIT THE KEY FOR YOUR CHOICE, 1-6)
```

Eamon - non-modal combat



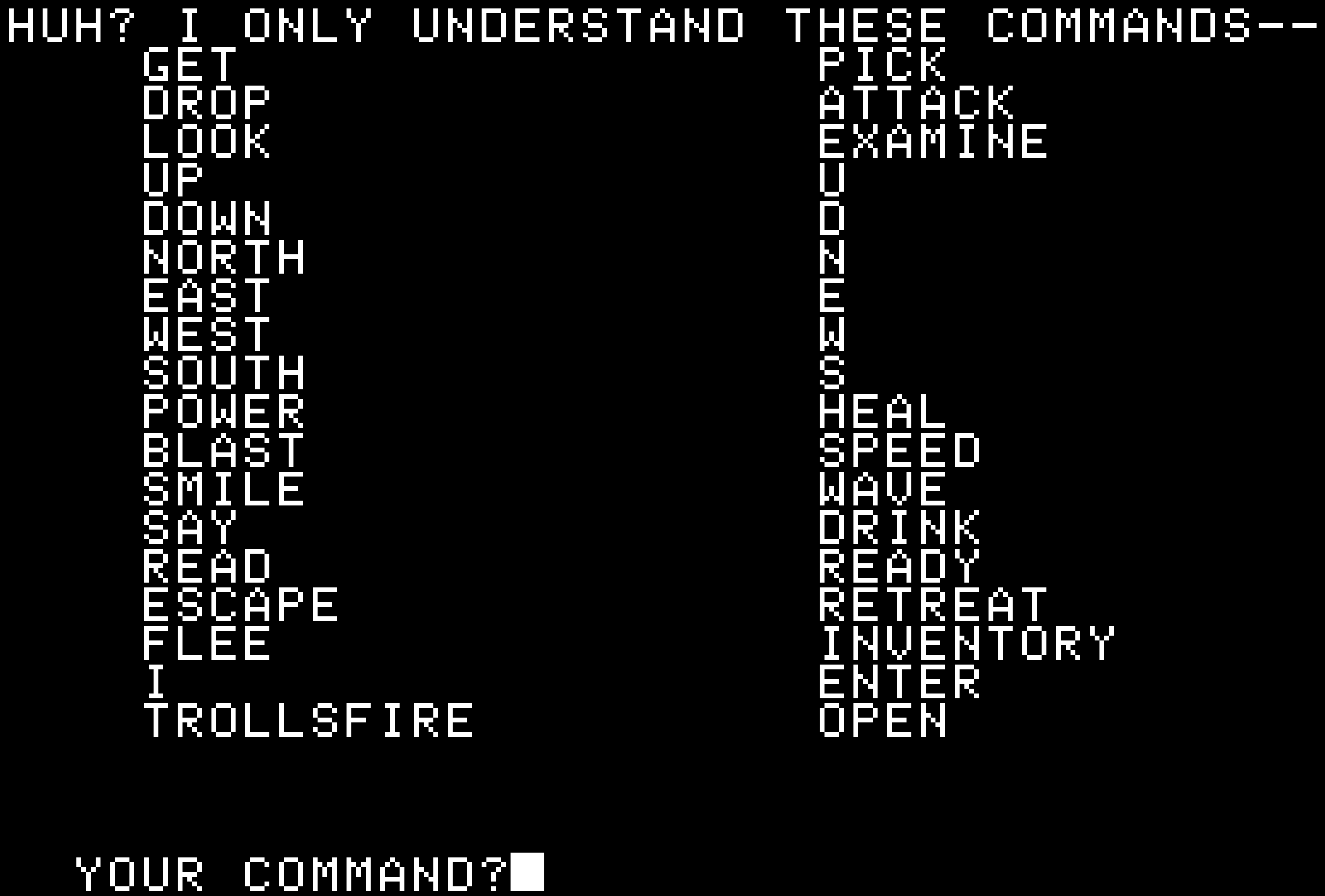
```
  YOUR COMMAND?ATTACK RAT
LEE ATTACKS TAN RAT
  --A MISS.
BLACK RAT ATTACKS LEE
  --A MISS.

BROWN RAT FLEES OUT AN EXIT.

TAN RAT ATTACKS HERMIT
  --A MISS.
HERMIT ATTACKS BLACK RAT
  --A HIT!

BLACK RAT IS IN PAIN.

YOU ARE STANDING IN
  THE WEST SIDE CHAMBER

BLACK RAT IS HERE.
TAN RAT IS HERE.
HERMIT IS HERE.
YOU SEE DIAMONDS

  YOUR COMMAND?█
```

Eamon stats



```
TAKE CARE OF YE.'



THE IRISHMAN SAYS, 'FIRST I MUST KNOW
WHETHER YE BE MALE OR FEMALE. WHICH ARE
YE  (HIT 'M' FOR MALE, 'F' FOR FEMALE)?M



THE IRISHMAN WALKS AWAY AND IN WALKS A
TALL MAN OF POSSIBLY ELVISH DESCENT.

HE STUDIES YOU FOR A MOMENT AND SAYS,
'HERE IS A BOOKLET OF INSTRUCTION FOR
YOU TO READ, AND YOUR PRIME ATTRIBUTES
ARE--

   HARDINESS--10
   AGILITY  --11
   CHARISMA --18'

(HIT 'R' TO READ INSTRUCTIONS, 'T' TO
   GIVE THEM BACK)█
```

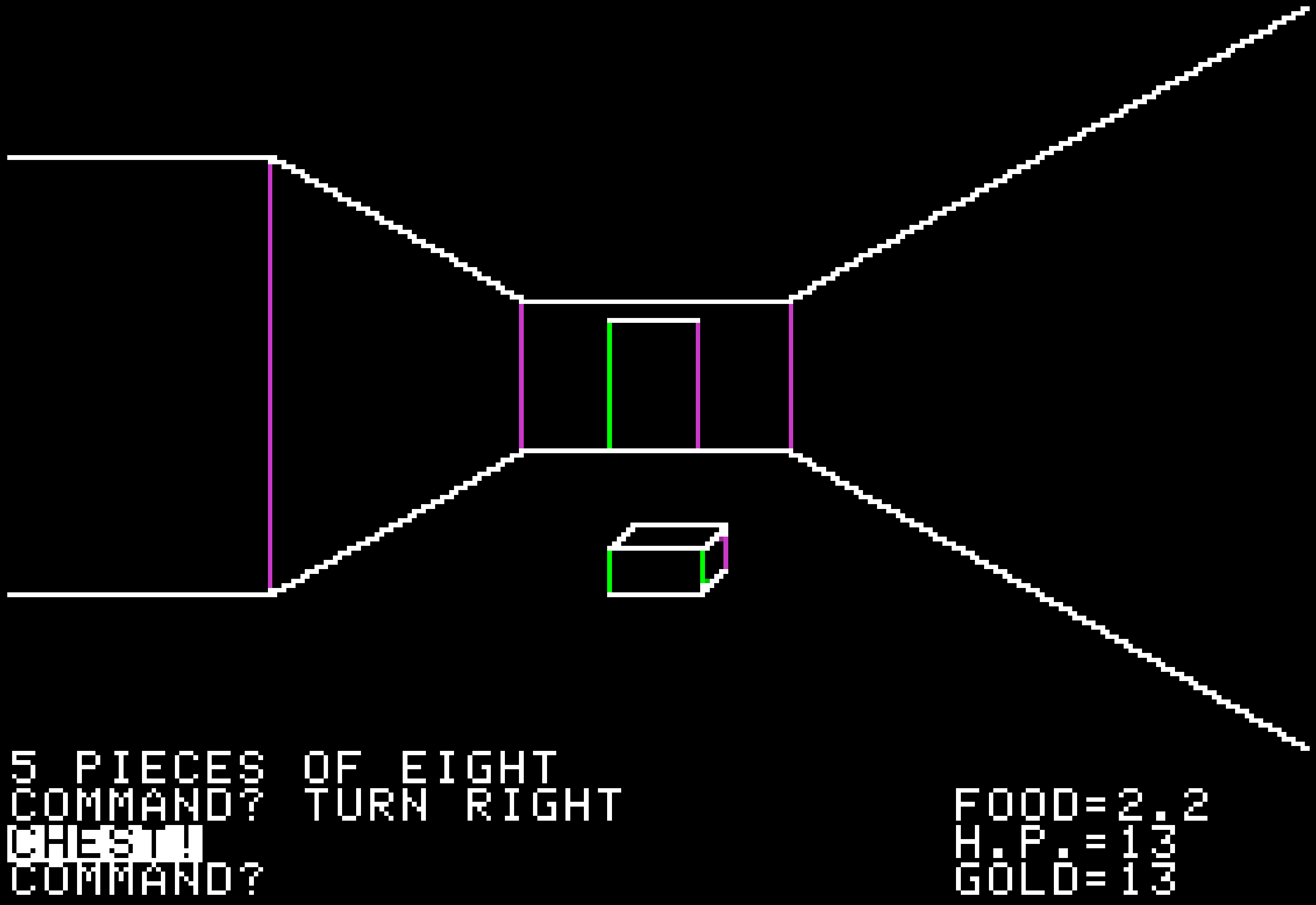
Eamon commands



```
HUH? I ONLY UNDERSTAND THESE COMMANDS--
    GET                 PICK
    DROP                ATTACK
    LOOK                EXAMINE
    UP                  U
    DOWN                D
    NORTH               N
    EAST                E
    WEST                W
    SOUTH               S
    POWER               HEAL
    BLAST               SPEED
    SMILE               WAVE
    SAY                 DRINK
    READ                READY
    ESCAPE              RETREAT
    FLEE                INVENTORY
    I                   ENTER
    TROLLSFIRE          OPEN



  YOUR COMMAND?█
```

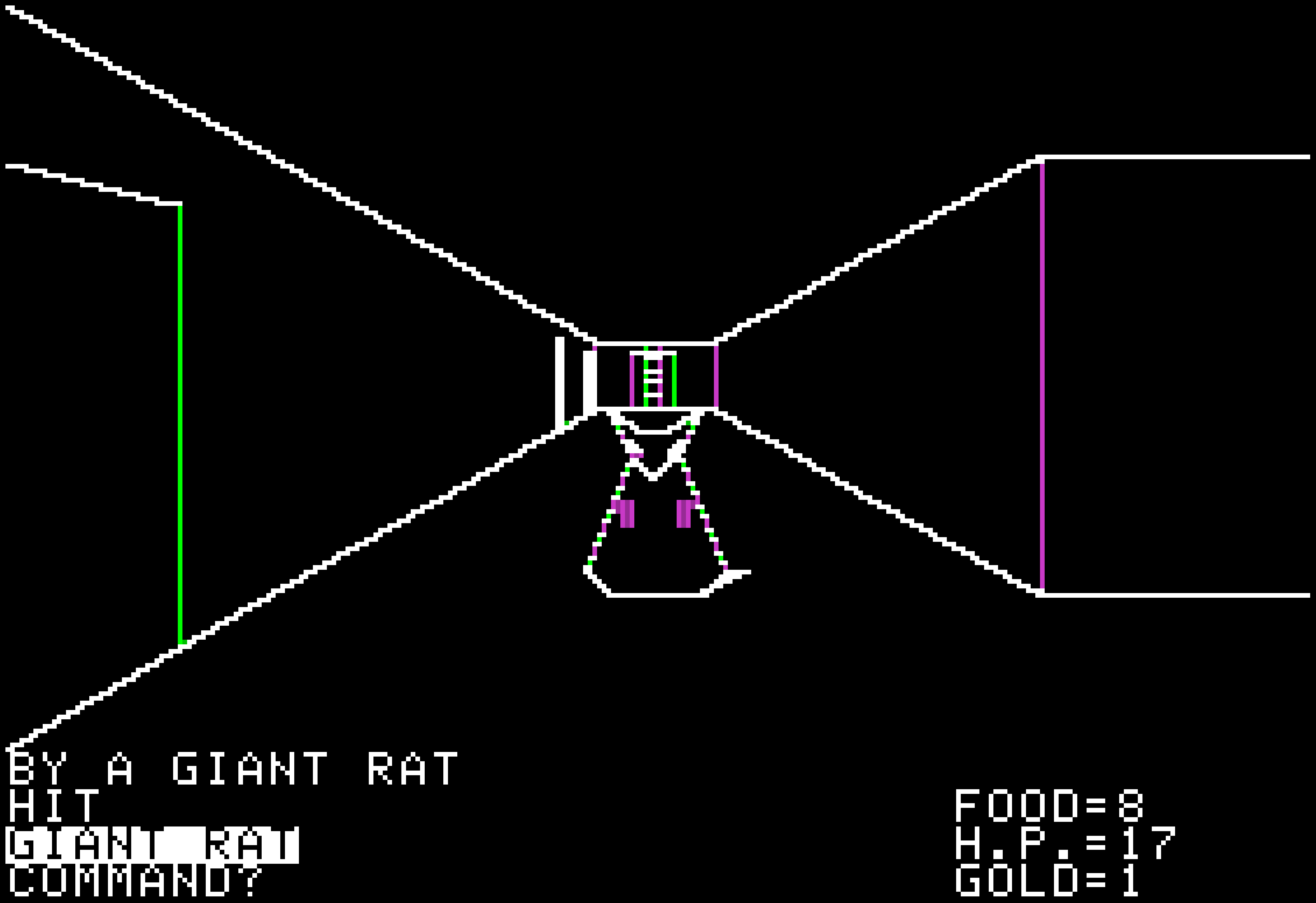
Akalabeth: World of Doom (1980)

by Richard Garriott



```
5 PIECES OF EIGHT
COMMAND? TURN RIGHT                         FOOD=2.2
CHEST!                                      H.P.=13
COMMAND?                                    GOLD=13
```

Akalabeth - non modal combat

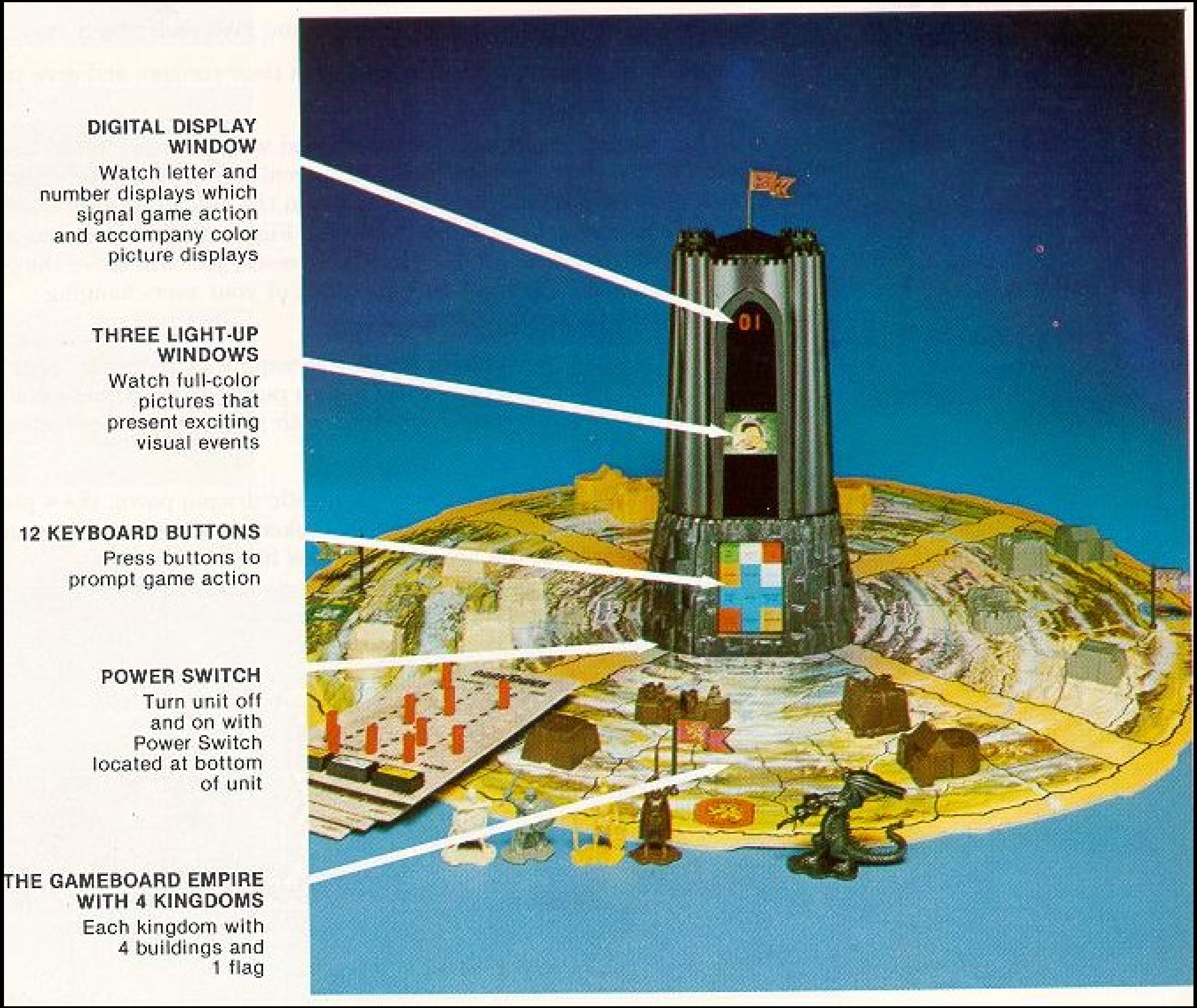


```
BY A GIANT RAT
HIT
GIANT RAT                            FOOD=8
COMMAND?                             H.P.=17
                                     GOLD=1
```

Dark Tower (1981)

Milton Bradley

2 D batteries, 12000 mAh



DIGITAL DISPLAY WINDOW
Watch letter and number displays which signal game action and accompany color picture displays

THREE LIGHT-UP WINDOWS
Watch full-color pictures that present exciting visual events

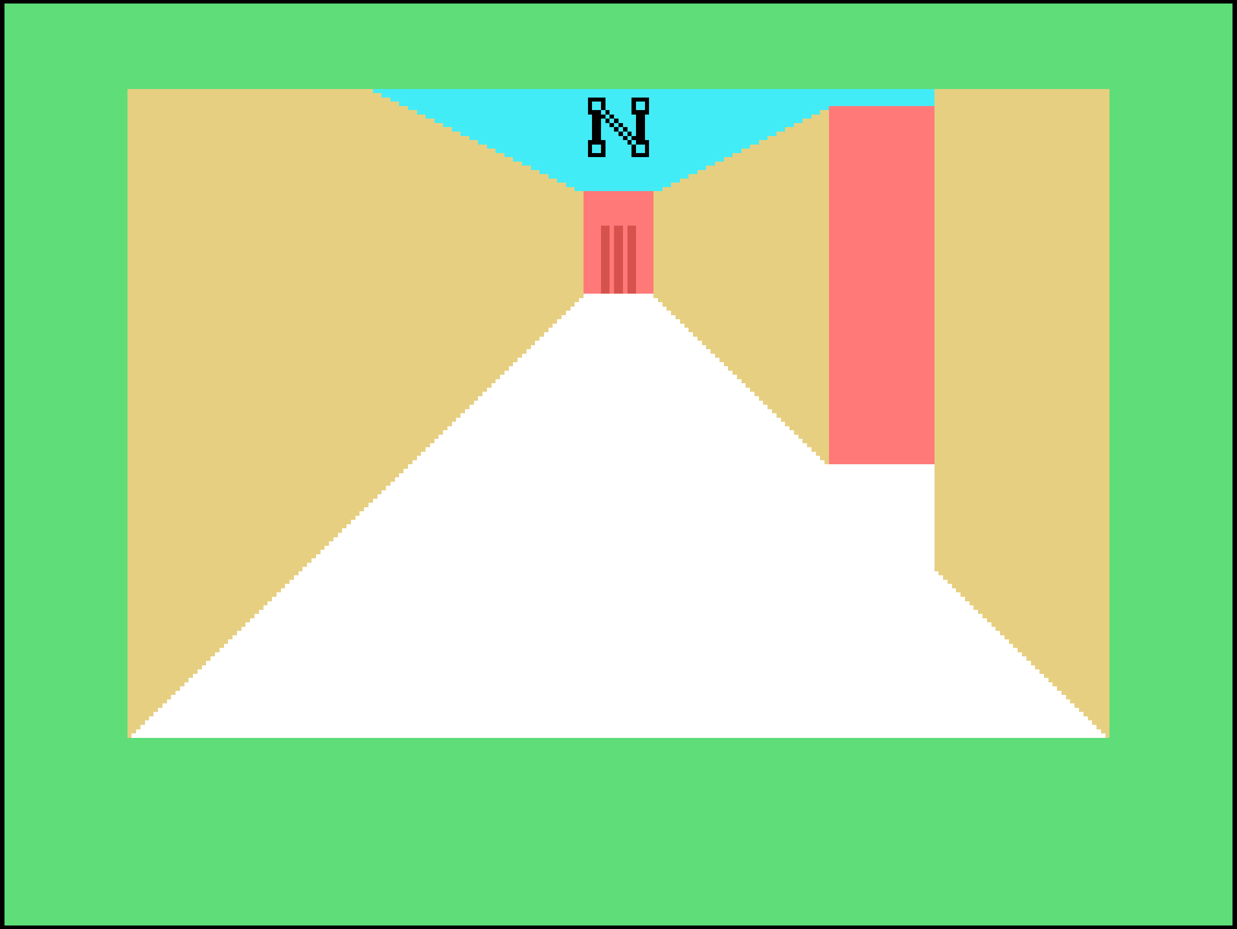
12 KEYBOARD BUTTONS
Press buttons to prompt game action

POWER SWITCH
Turn unit off and on with Power Switch located at bottom of unit

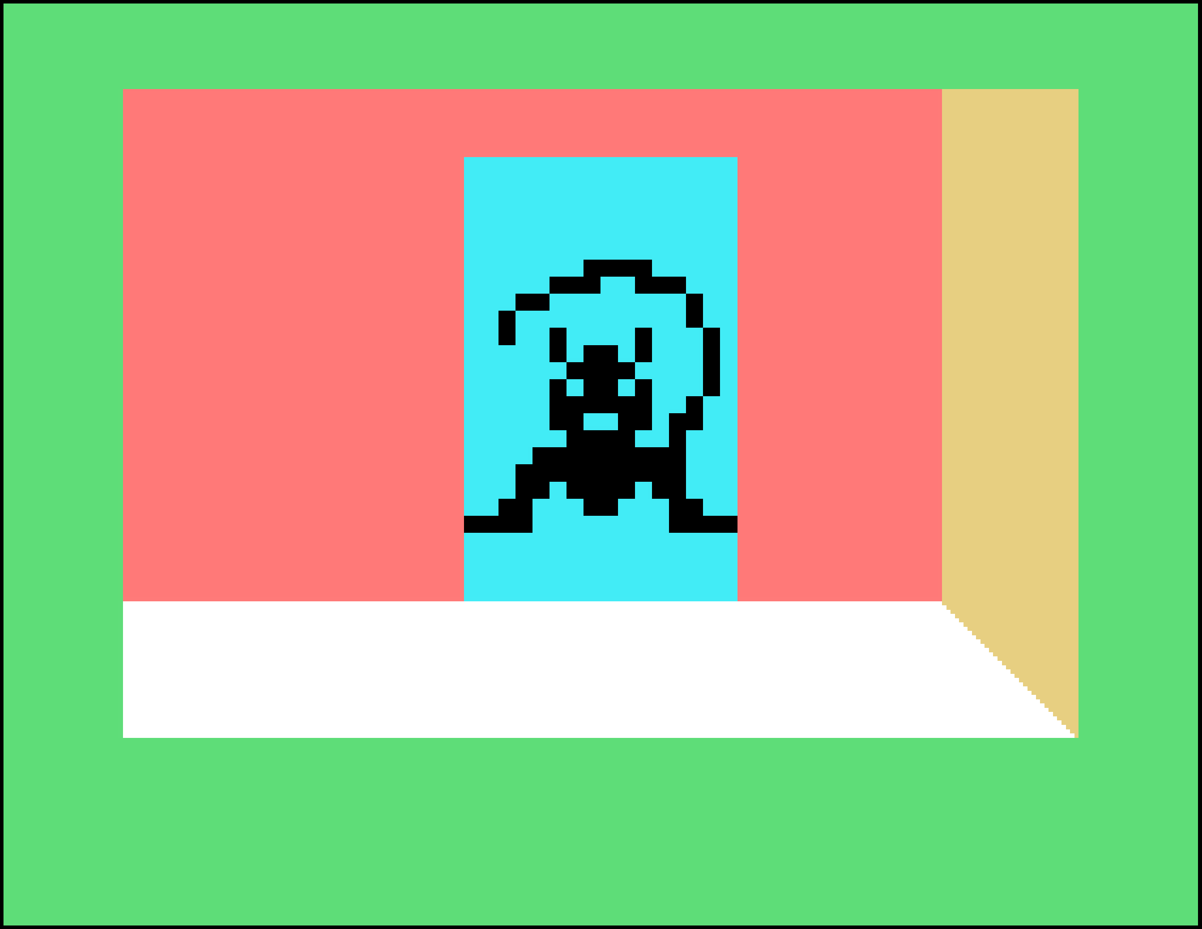
THE GAMEBOARD EMPIRE WITH 4 KINGDOMS
Each kingdom with 4 buildings and 1 flag

Tunnels of Doom (1982)

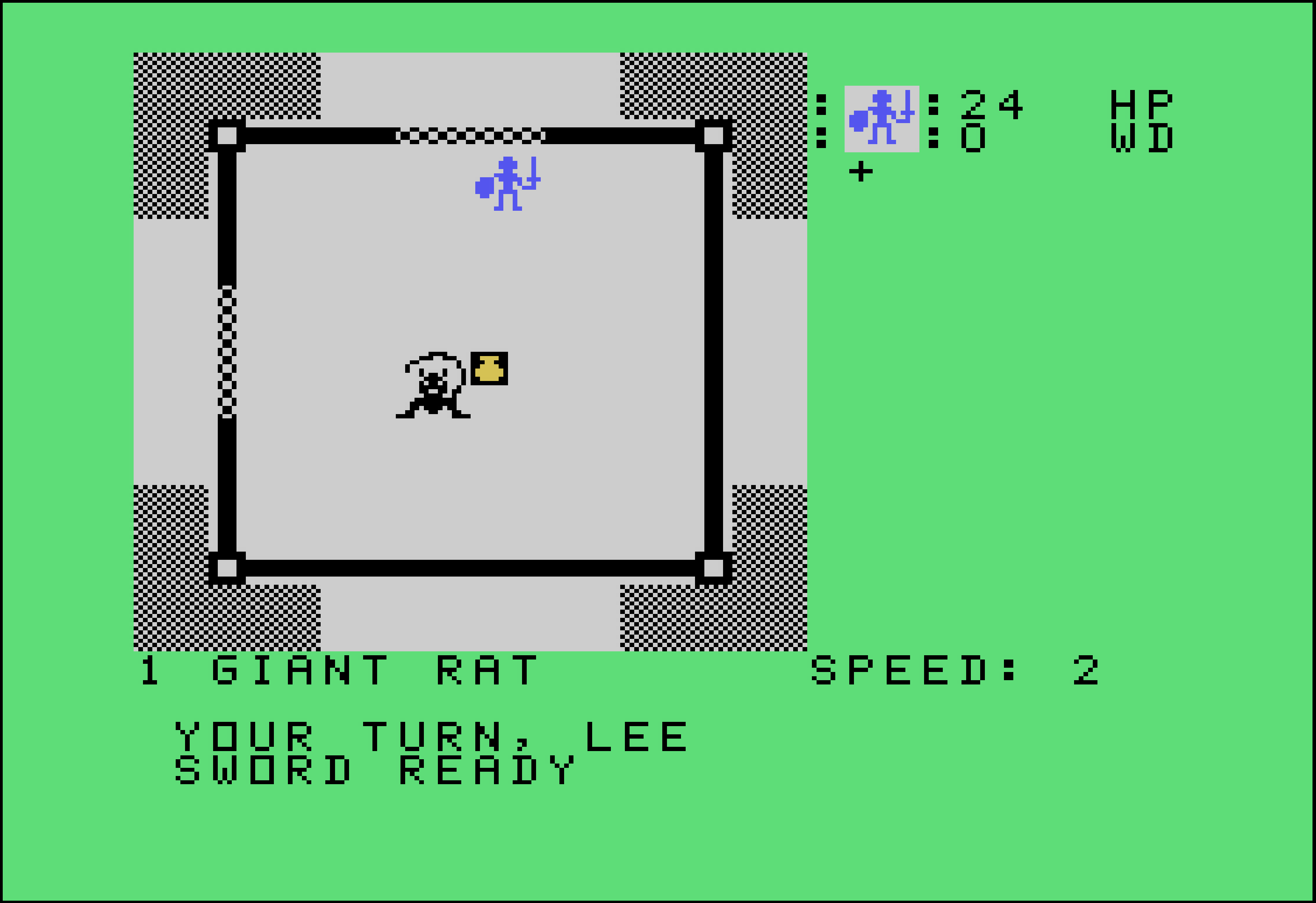
by Kevin Kenney



Tunnels of Doom - Giant Rat

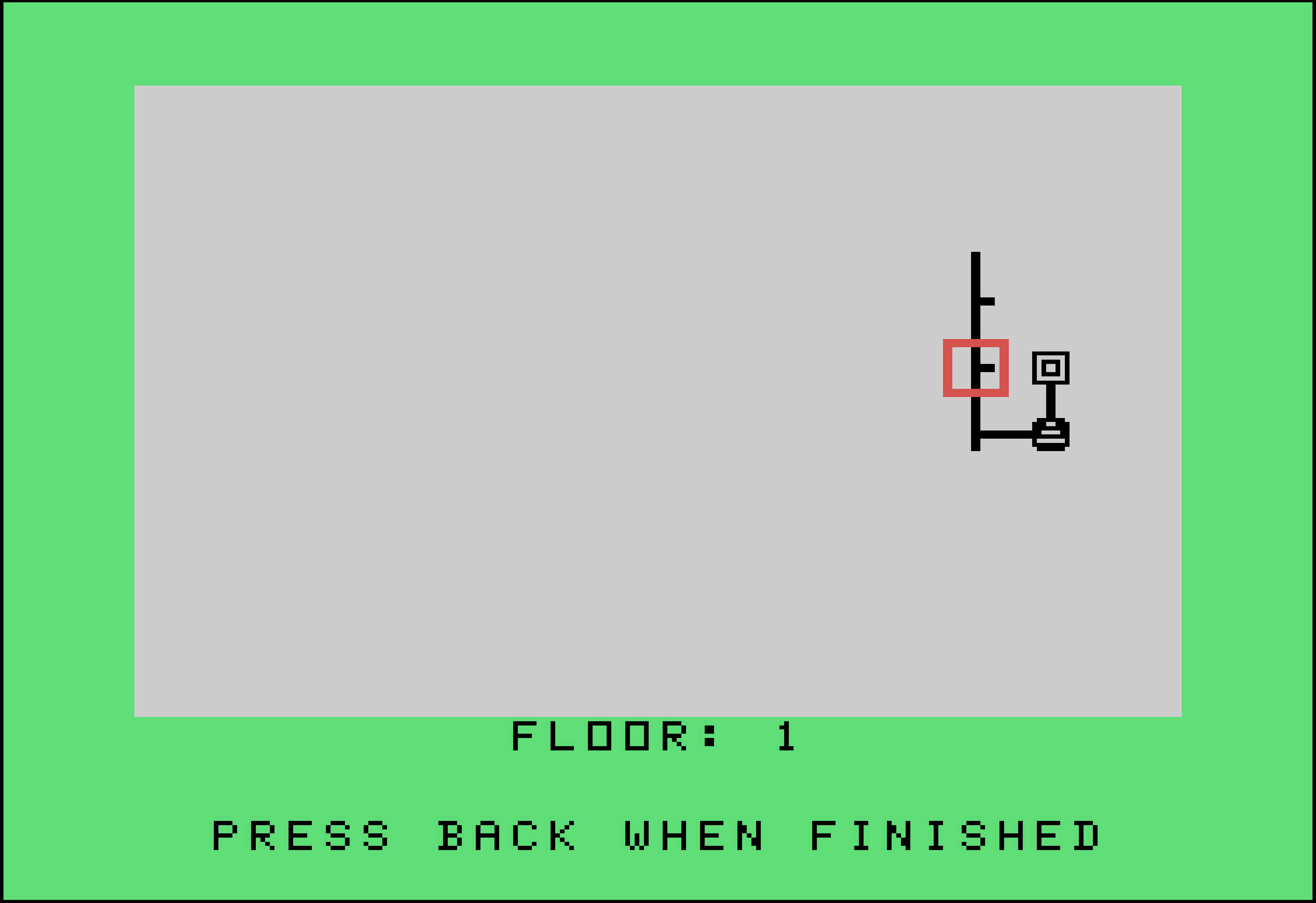


Tunnels of Doom - modal combat



```
:24    HP
:0     WD
+

1 GIANT RAT              SPEED: 2

 YOUR TURN, LEE
 SWORD READY
```

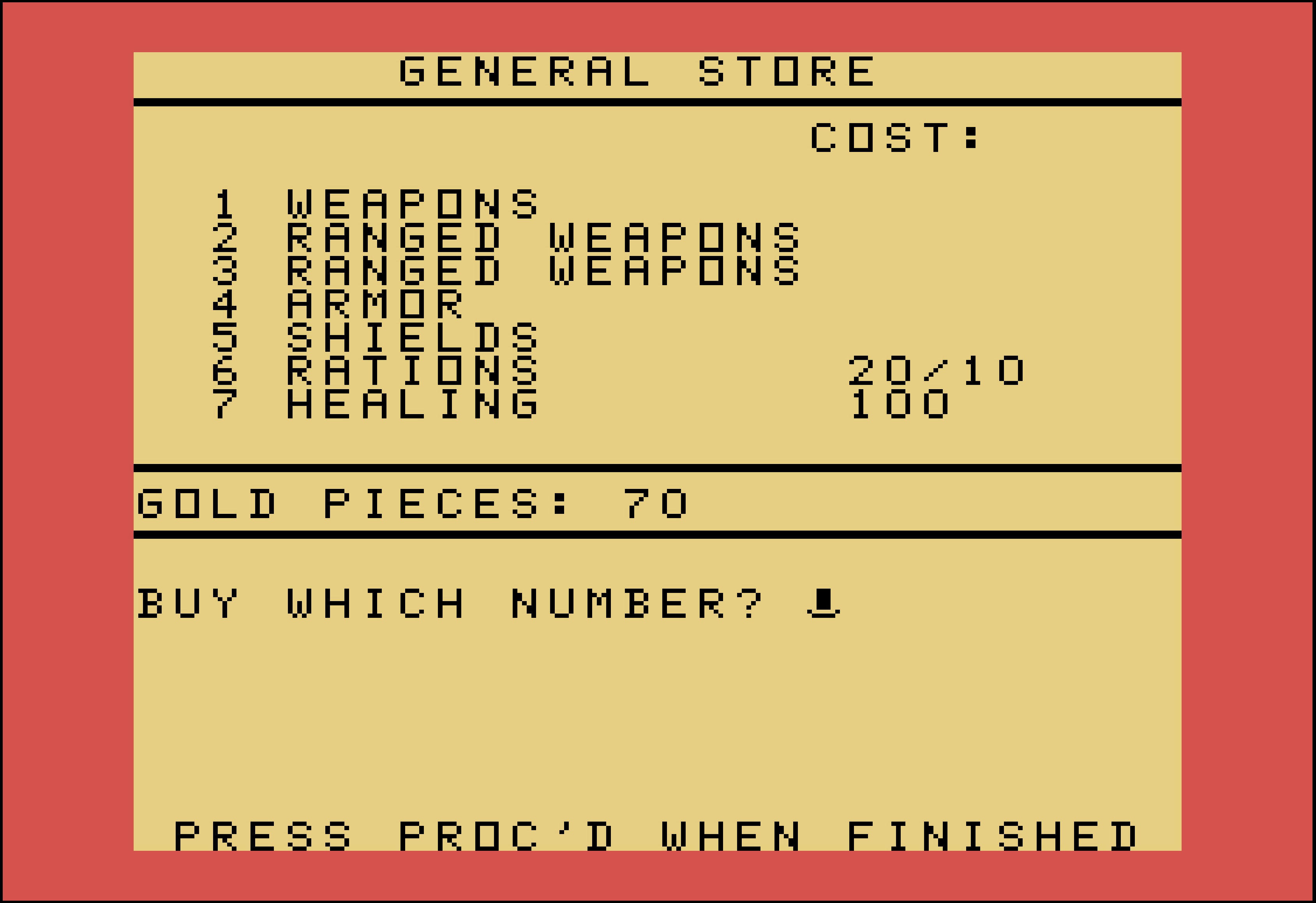
Tunnels of Doom overhead map



FLOOR: 1

PRESS BACK WHEN FINISHED

Tunnels of Doom general store

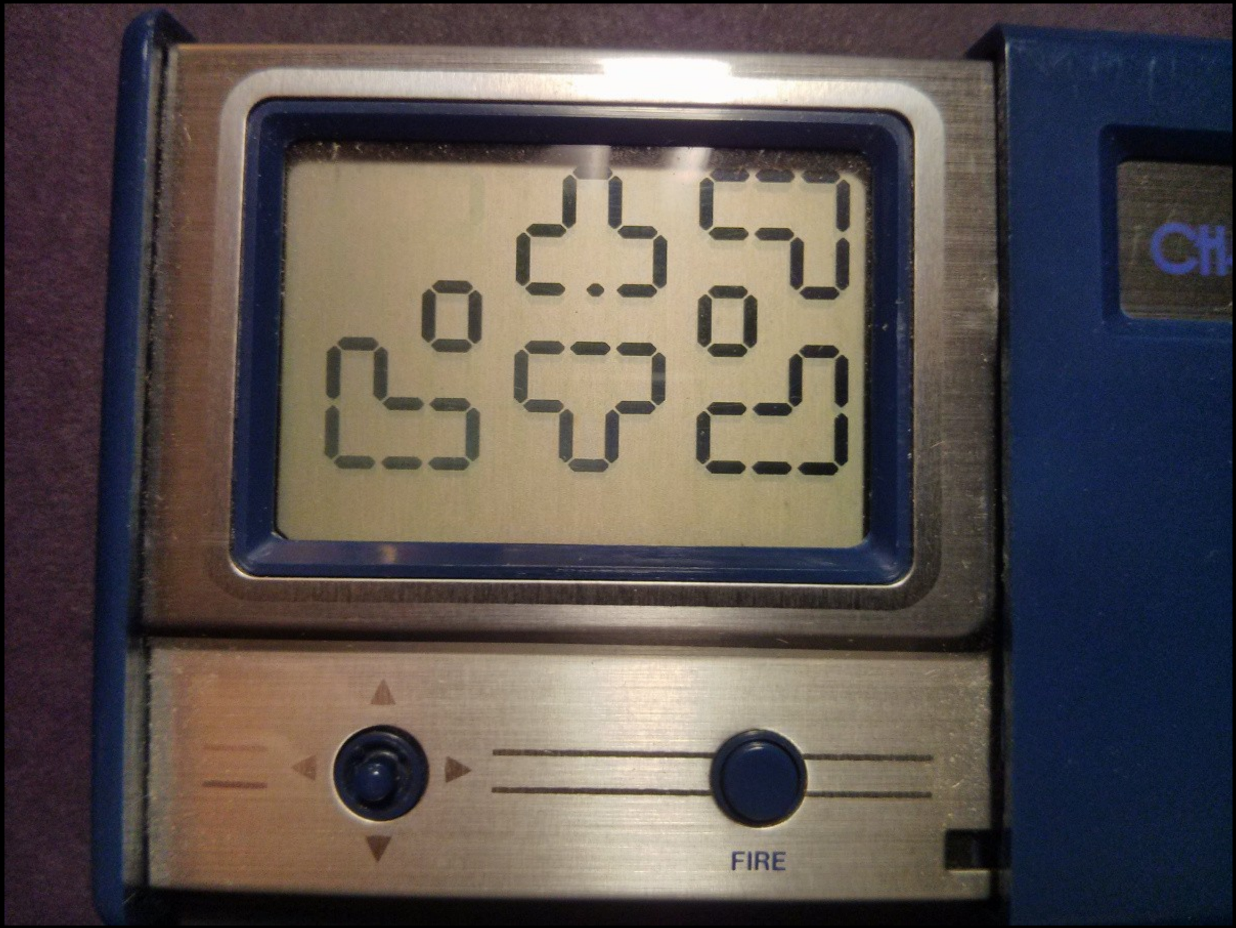


```
              GENERAL STORE
                                  COST:

    1  WEAPONS
    2  RANGED WEAPONS
    3  RANGED WEAPONS
    4  ARMOR
    5  SHIELDS
    6  RATIONS                      20/10
    7  HEALING                      100

GOLD PIECES: 70

BUY WHICH NUMBER?

 PRESS PROC'D WHEN FINISHED
```

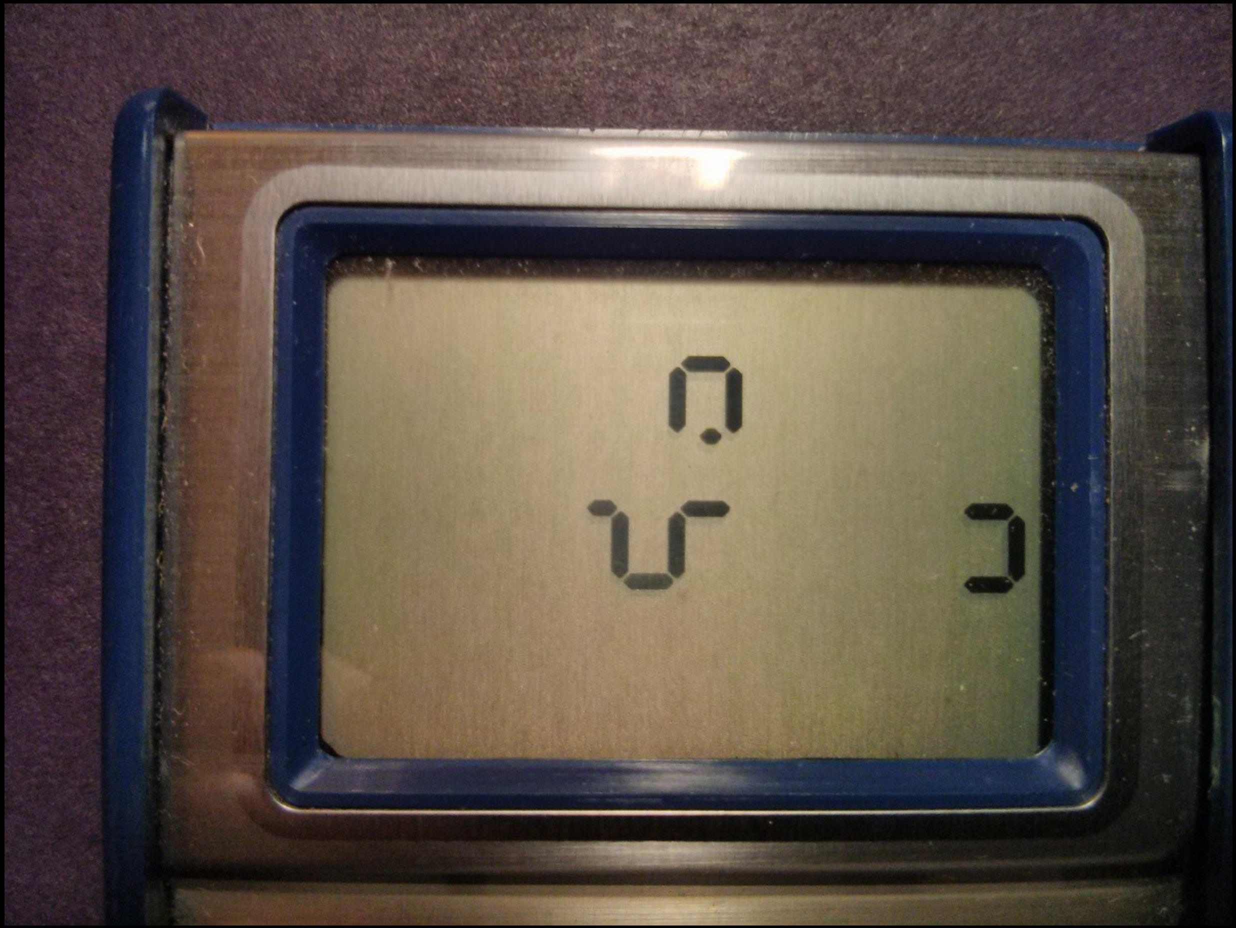
Chase-N-Counter, GCE (1982)

2 SR44 batteries (200 mAh)

Treasure Trek



Going after the treasure



Roguelike "core" in early games

- Rogue used ASCII as spatial symbols, overhead view
- Eamon used ASCII as pure English, to narrate the world
- Garriott's Akalabeth dungeons used 1st person perspective, rooms/halls single unit, one monster per unit
- Tunnels of Doom used 1st person, but combat took place in room with interior space larger than 1 unit
- Dark Tower used cardboard for the game map, the cpu keeping track of position. Not a dungeon crawl, but overhead map combat, like Ultima's land view
- Chase-N-Counter, used low-power LCD segments for game map, and had joystick.

Why a Microcontroller?

- The physical limitations of the early hardware actually added mystique to the games (slow map generation, visuals lacking detail or no visuals at all)
- In the early 1980's dungeon crawls, you treaded *slowly*, not as slow as a paper-based RPG, but much slower than a modern roguelike.
- Just staring at an empty hall or trying to figure out what to say to a parser is fascinating since *anything* is possible.
- It is a small physical "object", and ties you to the physical world (like Garriott's peculiar obsession with having working chairs in his MMORPGs)

- Being small and self-contained with a hidden binary (lock bits), a it is an ideal "magical box" with secrets only revealed to the determined adventurer.

- The roguelike doesn't just have to have permadeath, it can also incorporate perma-end-of-world, since the game can only be played once if there is no internal procedural generation and the EEPROM is set to self-destruct.

- They are cheaper and less complicated than FPGAs and have extremely low power requirements (more on this later).

- On a modern computer when you add more features, the program size gets larger and more inefficient. On a microcontroller, as you add more features, the program size remains the same, but becomes more efficient. The act of programming becomes an adventure in itself, sort of like a "Quest to reach the Landauer Limit".

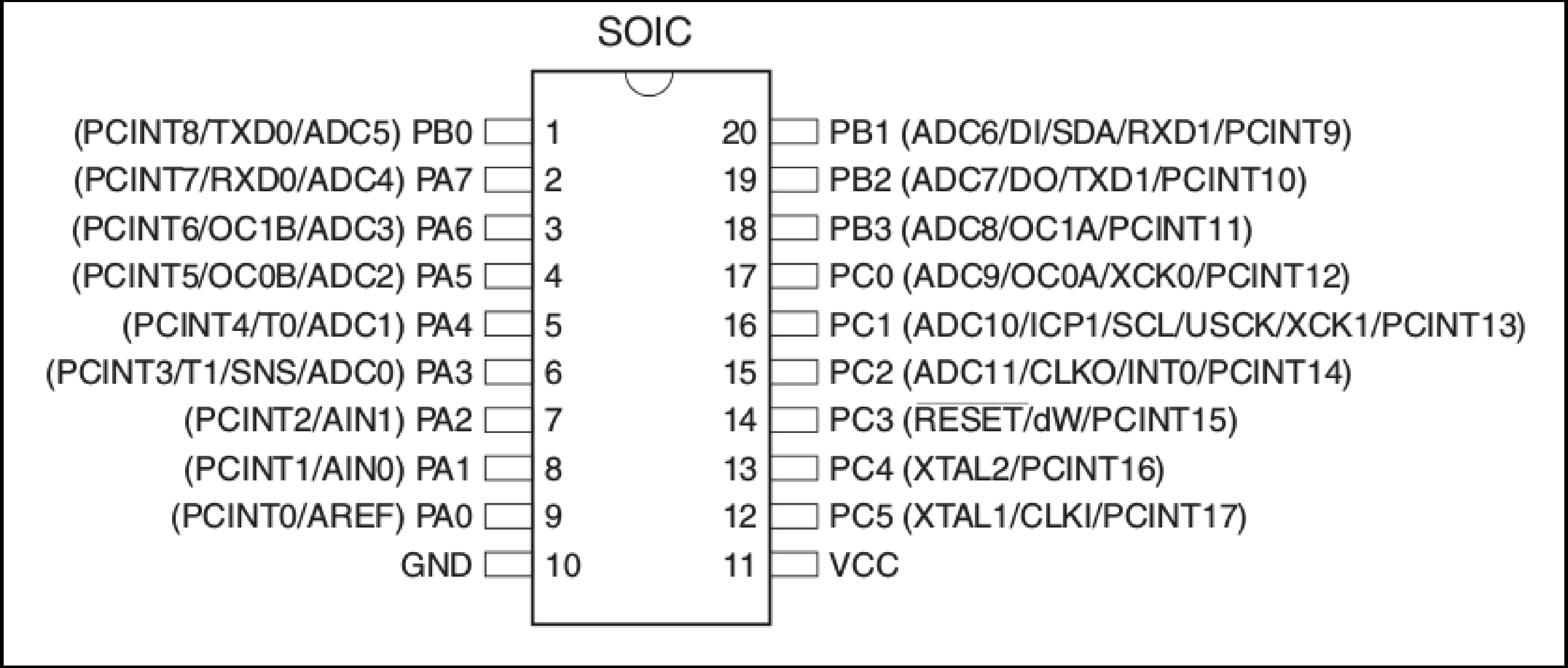
Why an AVR?

- It is well-supported by the open-source GCC complier.
- The popularity of the Arduino, being AVR-based, means that there is a lot of "indirect" information about AVR microcontrollers if you need it.

Why ATTiny 1634?

- It comes with 16 KB flash (the more, the better)
- It includes a *very* low power mode
- It has a hardware UART (2 actually)
- The *same* MISO/MOSI lines also function as a UART.
- It has capacitive touch (but I decided not to use it)
- It is inexpensive (under $2 in the US)
- No additional programmer is needed. A $20 Raspberry Pi A+ will work as both a programmer and a development workstation.

Atmel ATTiny 1634


SOIC
(PCINT8/TXD0/ADC5) PB0 1
(PCINT7/RXD0/ADC4) PA7 2
(PCINT6/OC1B/ADC3) PA6 3
(PCINT5/OC0B/ADC2) PA5 4
(PCINT4/T0/ADC1) PA4 5
(PCINT3/T1/SNS/ADC0) PA3 6
(PCINT2/AIN1) PA2 7
(PCINT1/AIN0) PA1 8
(PCINT0/AREF) PA0 9
GND 10
20 PB1 (ADC6/DI/SDA/RXD1/PCINT9)
19 PB2 (ADC7/DO/TXD1/PCINT10)
18 PB3 (ADC8/OC1A/PCINT11)
17 PC0 (ADC9/OC0A/XCK0/PCINT12)
16 PC1 (ADC10/ICP1/SCL/USCK/XCK1/PCINT13)
15 PC2 (ADC11/CLKO/INT0/PCINT14)
14 PC3 (RESET/dW/PCINT15)
13 PC4 (XTAL2/PCINT16)
12 PC5 (XTAL1/CLKI/PCINT17)
11 VCC

Time/Space/Speed Comparison

Approximated typical specs (kB and Mhz)

- A modern PC has approximately 8 million times the RAM, 400 times the clock cycles, 60 million times the storage of an ATTiny1634.

- Modern PC: 8,000,000K RAM, 1,000,000,000K HD, 3000+ Mhz
- Modern smartphone: 3,000,000K RAM, 32,000,000K Flash, 2000 Mhz
- Raspberry Pi A+: 256,000K RAM, 32,000,000K Flash, 700+ Mhz
- Commodore 64: 64K RAM, ~4-16K ROM Cart, 170K Floppy, 1 Mhz
- Apple II: 48K RAM, 140K Floppy, 1 Mhz
- TI-99/4A: 16K RAM, ~8-32K ROM Cart, 92K Floppy, 3 Mhz
- ATtiny1634: 1K RAM, 16K Flash, 0.25K EEPROM, 8 Mhz (int. oscillator)

But...

Atari 2600 (1977)

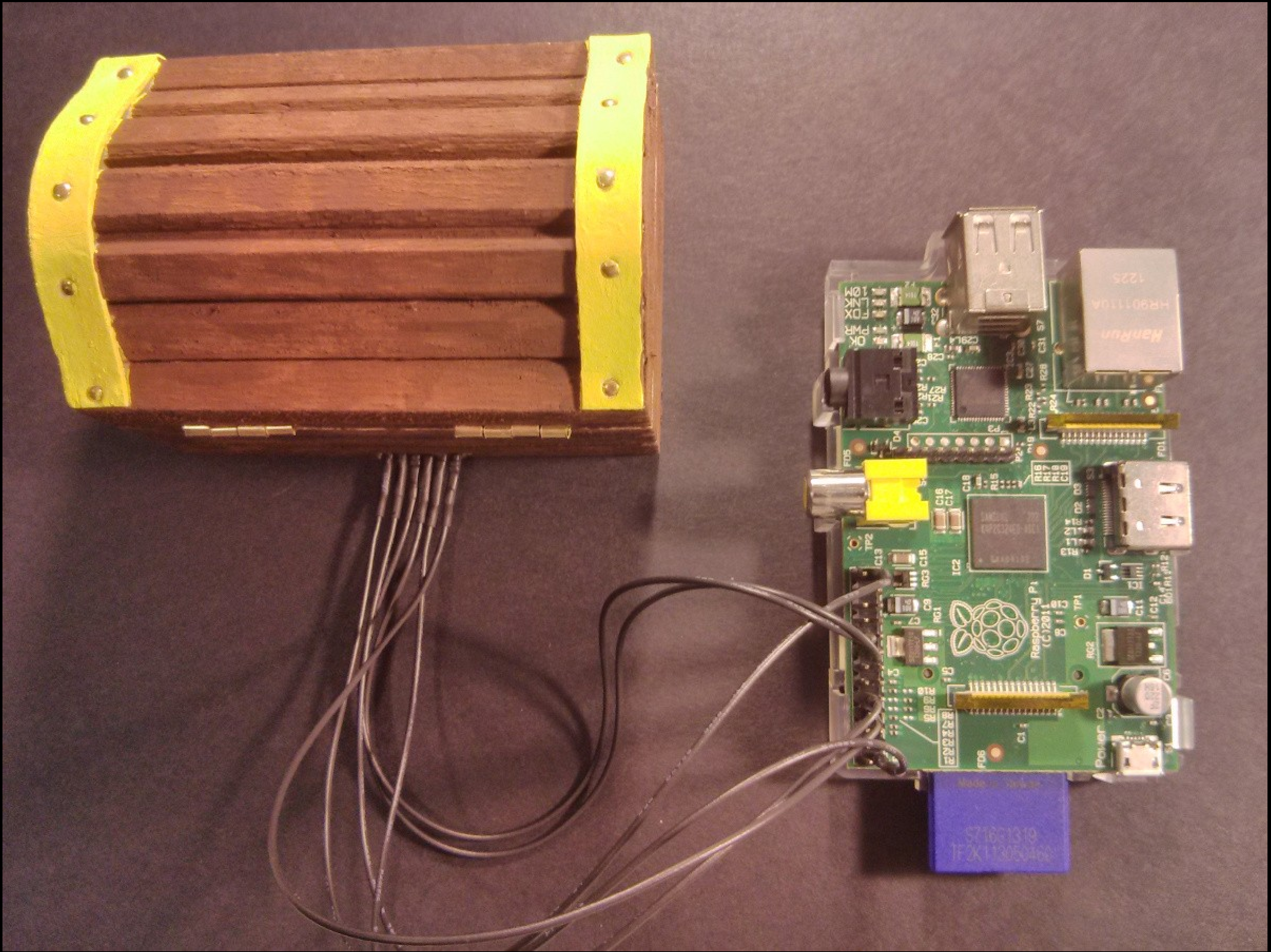
- It had 1/8th the resources of an ATTiny1634!
- 128 bytes RAM. BYTES!!
- 2 to 4K Cartridge ROM!!
- But they still made very entertaining games!
- If the Atari programmers could do it in Assembly in the 1970's, then *surely* an entertaining roguelike can be created in C language with 8 times the RAM, ROM, and speed!

Development environment

- Hardware: Raspberry Pi, ATtiny1634
- Software: Arch Linux, avr-gcc, avr-binutils, avr-libc, avrdude, wiringpi, minicom, text editor
- Remove ttyAMA0 lines from /boot/cmdline.txt and disable systemd serial_getty@ttyAMA0.service to prevent Arch Linux serial conflict
- Turn off hardware flow control in minicom, enable wordwrap, set baud rate to 9600
- Cross-connect the Raspberry Pi's UART lines Tx, Rx to the Rx, Tx lines on the AVR, so the Pi can communicate with the AVR immediately after programming since its SPI MOSI, MISO ports conveniently uses the same lines.
- GPIO jumper wires work without resistors since everything is 3.3 volt logic (Just make sure NOT to send an output high over SPI lines while connected.)

Getting Avrdude to work correctly

- Recompile Avrdude with "--enable-linuxgpio=yes" option. This can be done easily using the ABS (Arch Build System) and editing the PKGBUILD file.
- Edit /etc/avrdude.conf to enable linuxgpio programmer using GPIO lines 14,15 for MOSI,MISO respectively.
- There is a bug in Avrdude 6.1 causing ATTiny1634 flash memory errors when using linuxgpio. See http://savannah.nongnu.org/bugs/?40144 for workaround and edit /etc/avrdude.conf to fix.
- Avrdude 6.1 linuxgpio does not return the GPIO pins back to their previous functions which breaks the UART after it finishes programming. Use Wiringpi to return GPIO pins 14, 15 back to their original UART mode (ALT0).



Power

- ATtiny1634 can run up to 8Mhz on an internal oscillator using only a 3 volt CR2032 battery and uses only about 500 nanoamps when powered down waiting for interrupt (event)
- A 225 mAh CR2032 losing .5 uA would take about 36 years to deplete! The shelf life of the battery is usually less than 10 years.
- What if...

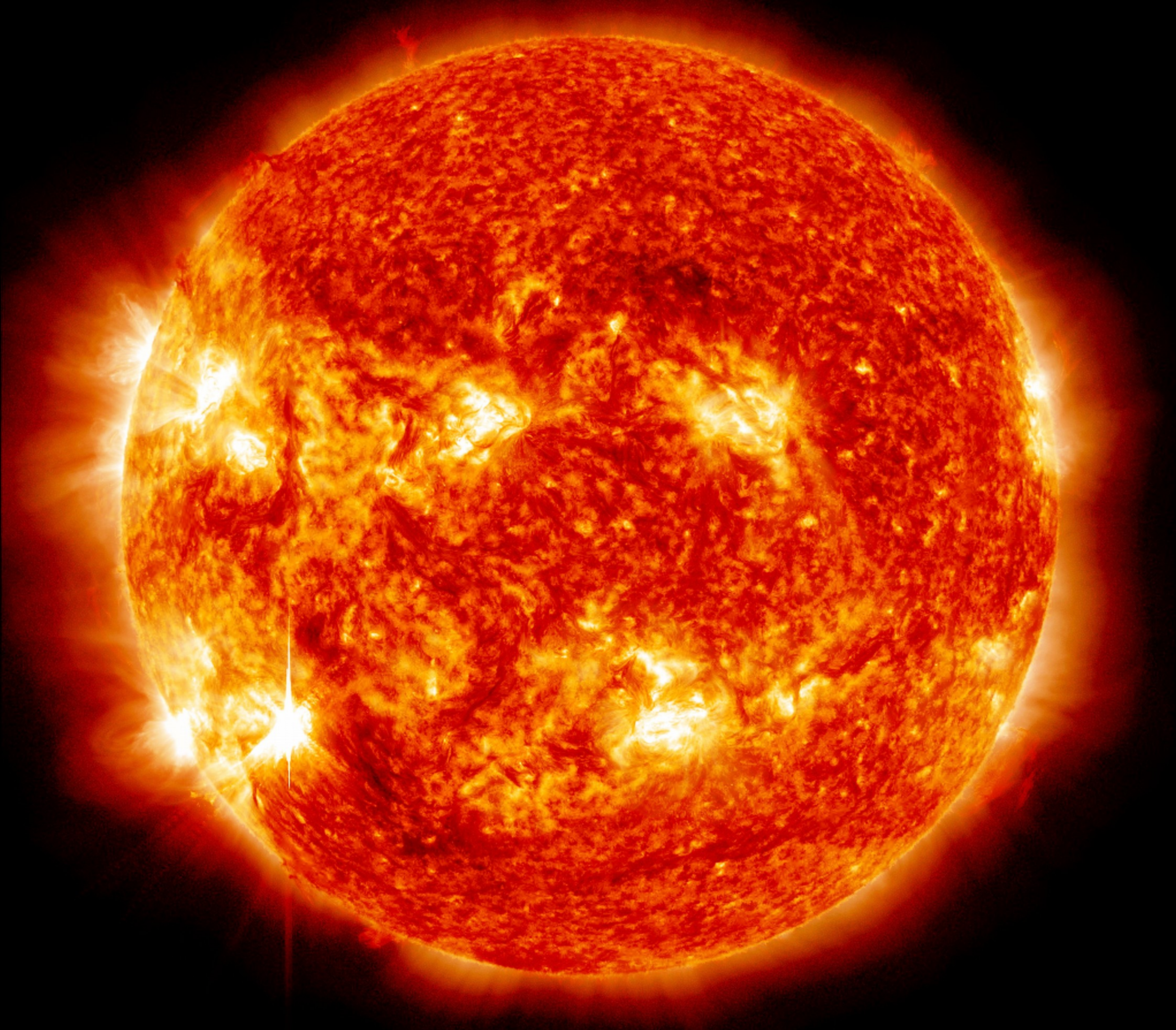
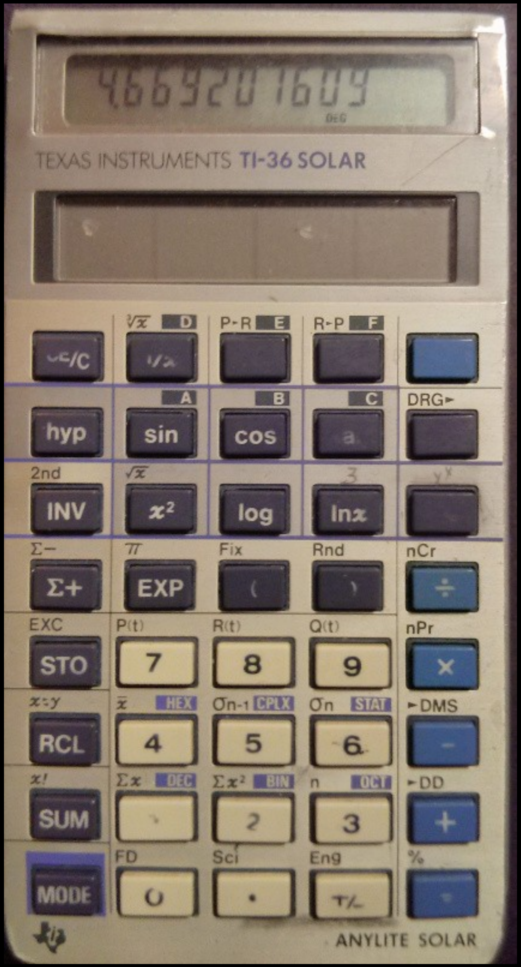
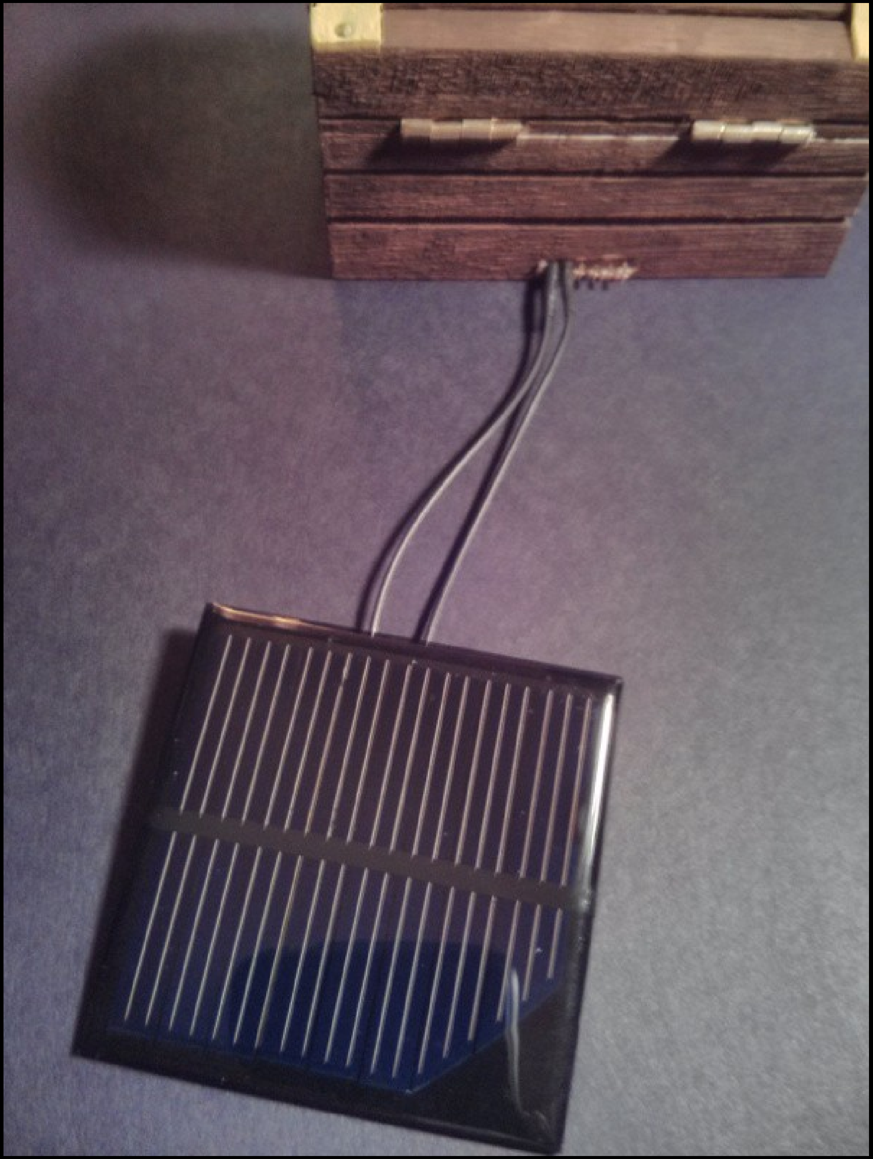


Image credit: NASA/SDO

TI-36 Solar Calculator (1989)

Zero batteries





A Solar Roguelike

- The ATTiny 1634 can easily run off of a $7 3.6 volt 100mA monocrystalline solar cell, drawing around 3 mA in Active mode (and low power LED's and piezo elements can draw < 2mA)
- Indoor lighting can produce enough current to power the device.
- The data retention of the device at room temperature is approximately 100 years and the solar cell may exceed 50 years.
- Internal EEPROM (256 bytes) is barely large enough to save game data in case of power failure (darkness), allowing 100,000 write cycles.
- Chip can be set for brown-out detection to auto-reset if voltage gets too low, preventing data corruption/malfuction.

C Language

- C Language is pretty much required (no objects)
- C++ can be used with AVR-GCC, but is incomplete and restricted.
- Interpreted languages like Python or BASIC are too large to fit on the Attinys, and using Assembly for a roguelike is very difficult.

User Interface options

- Create a dedicated electronic UI (Capacitive touch is possible but may not be ideal)
- Create a physical game board
- Create circuitry to interface game controllers and/or monitor
- Some people have generated NTSC and PAL composite video using simple circuits
- Turn the AVR into a UART server

Numeral systems for representation

- By selecting and balancing a positional numeral system, you can express the same information in different ways on a game board.
- Take the number of distinct numerals that you want to use (the base) and raise that to the power of the number of positions you want to use to determine the largest number you can represent.
- The Ancient Sumerians, for example, used base 60. This is likely why we have 60 minutes in a hour, 60 seconds in a minute, 360 degrees in a compass (60 x 6), and why we have a 12-hour clock and 12-sign zodiac (60 / 5).

International Morse Code

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

- It is not a binary code but can be converted to binary using padding
- Base 26 can be achieved with just A-Z.
- Timing can be difficult to accurately encode/decode. It is better to use separate circuits for dits and dahs or emulate a horizontal electronic keyer.
- Sound makes use of our auditory memory to greatly improve Morse recognition. It works surprisingly well for human beings, being designed in the 1840's.

Sometimes Morse can mimic the actual action

Hit

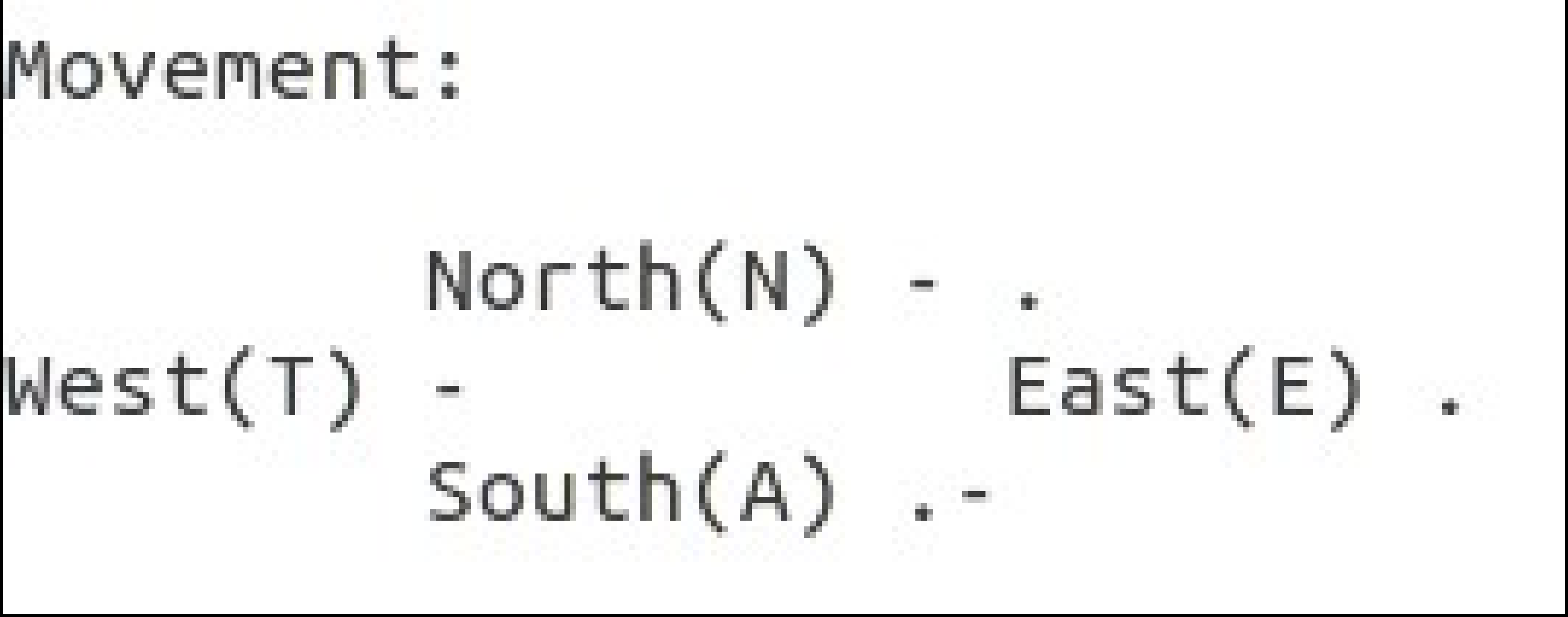
(H)

. . . .

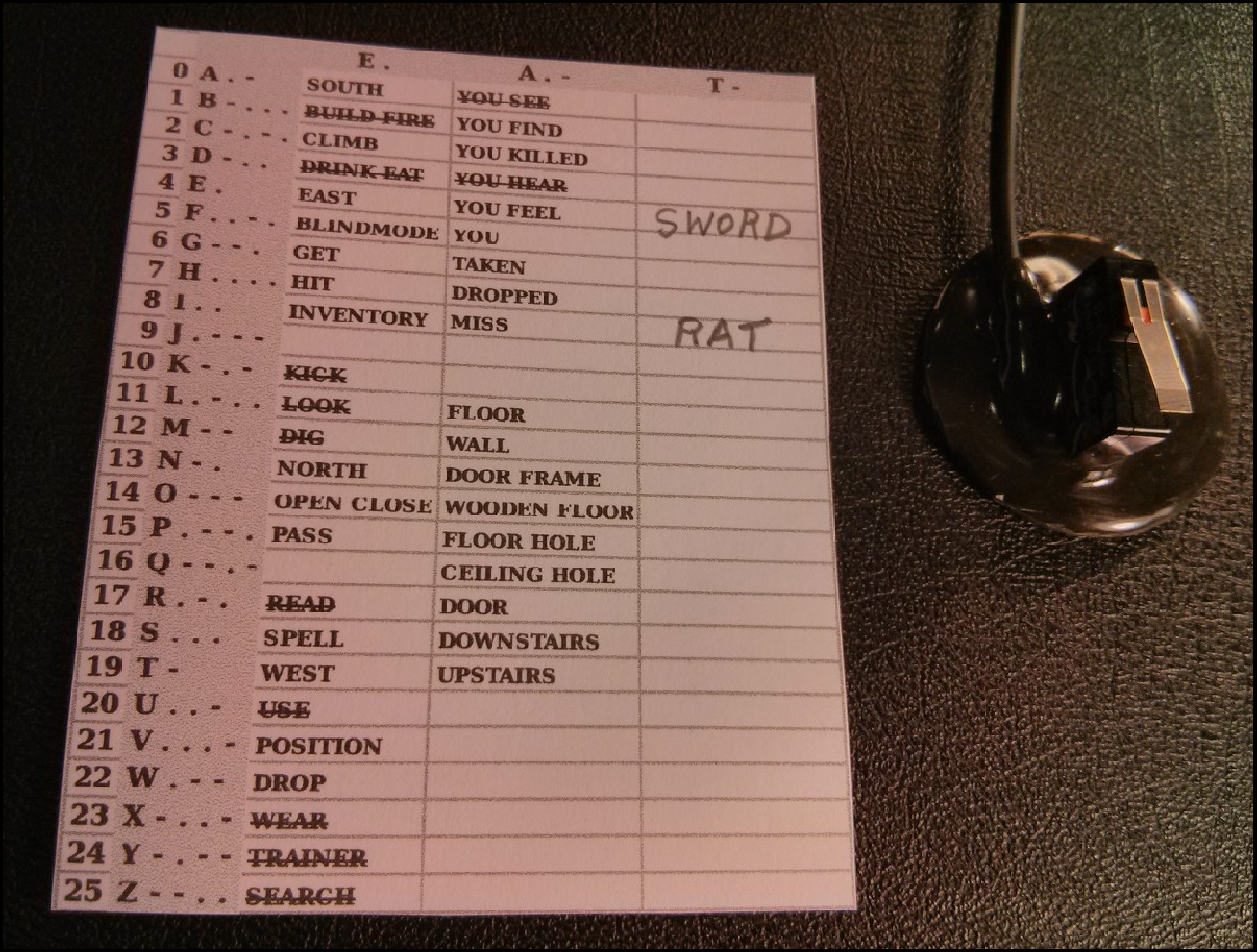
dit,dit,dit,dit

(like pounding your fist)

Frequent commands can use NEAT



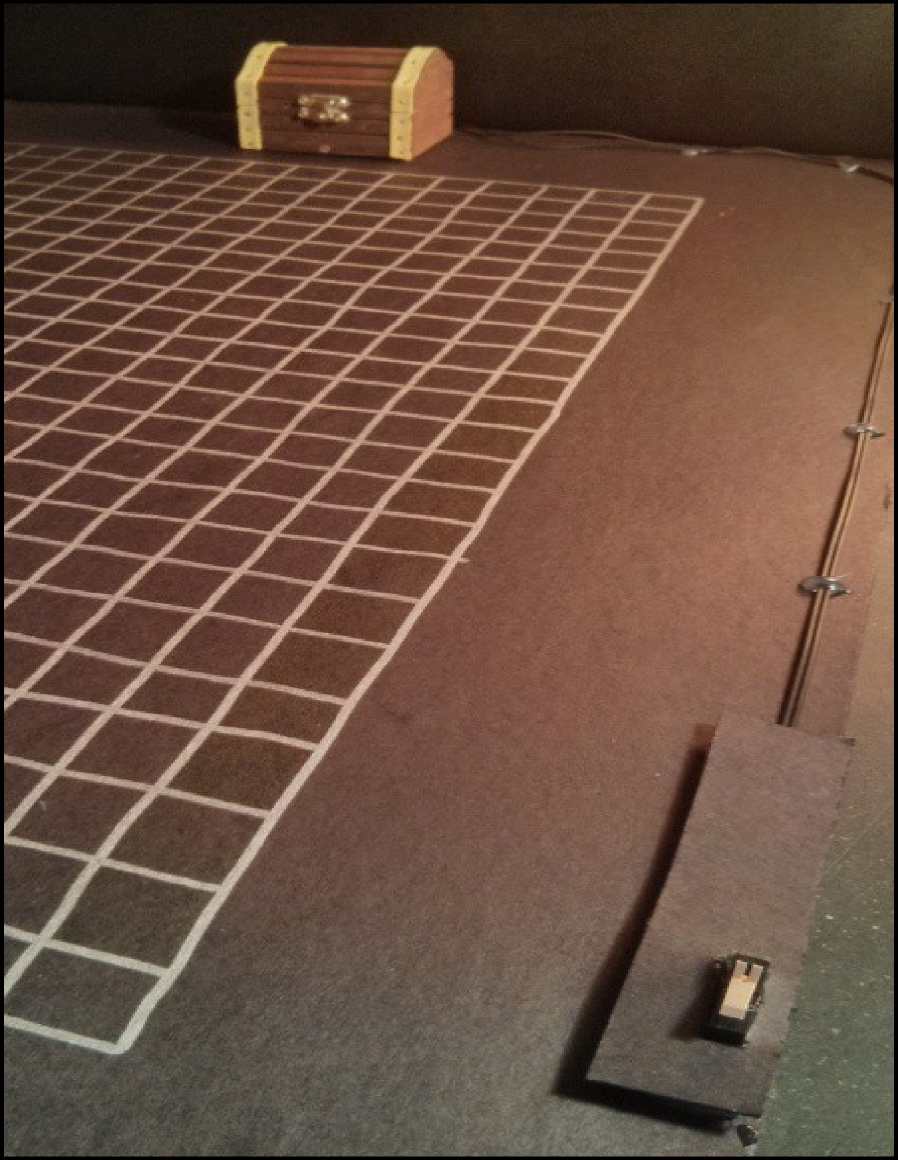
```
Movement:

        North(N) - .
West(T) -          East(E) .
        South(A) .-
```

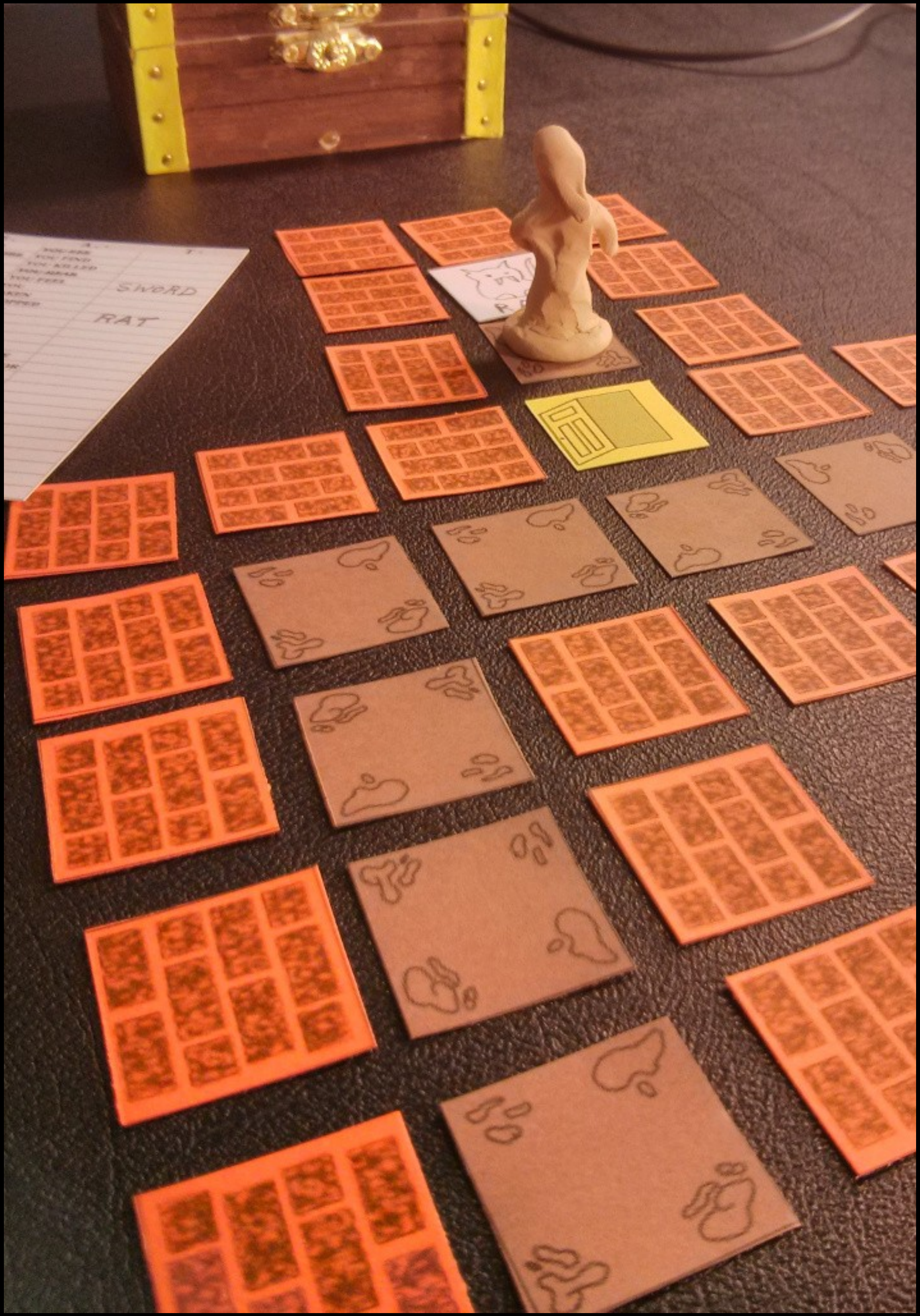


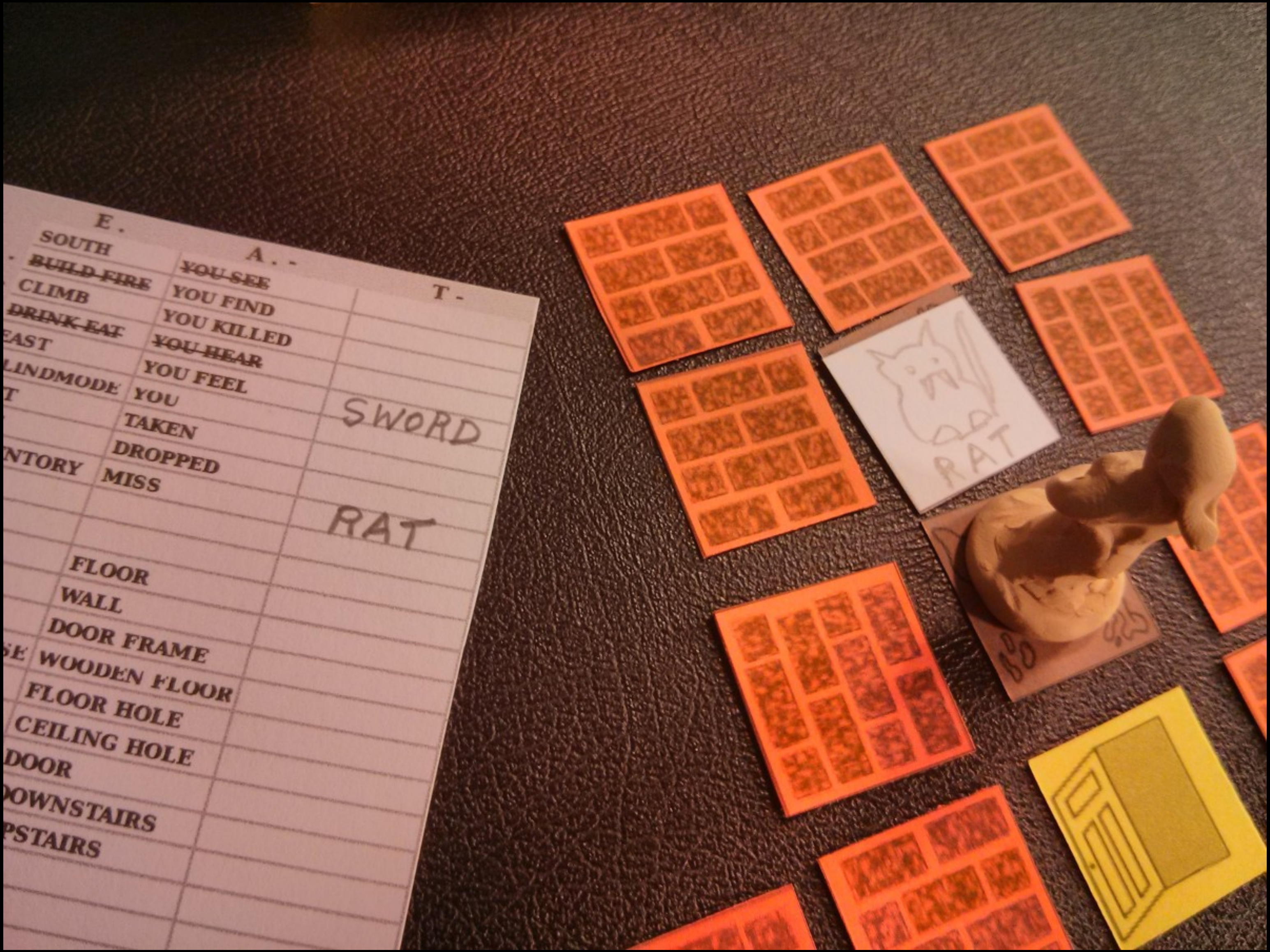
	E .	A . -	T -
0 A . -	SOUTH	~~YOU SEE~~	
1 B - . . .	~~BUILD FIRE~~	YOU FIND	
2 C - . - .	CLIMB	YOU KILLED	
3 D - . .	~~DRINK EAT~~	~~YOU HEAR~~	
4 E .	EAST	YOU FEEL	SWORD
5 F . . - .	BLINDMODE	YOU	
6 G - - .	GET	TAKEN	
7 H	HIT	DROPPED	
8 I . .	INVENTORY	MISS	RAT
9 J . - - -			
10 K - . -	~~KICK~~		
11 L . - . .	~~LOOK~~	FLOOR	
12 M - -	~~DIG~~	WALL	
13 N - .	NORTH	DOOR FRAME	
14 O - - -	OPEN CLOSE	WOODEN FLOOR	
15 P . - - .	PASS	FLOOR HOLE	
16 Q - - . -		CEILING HOLE	
17 R . - .	~~READ~~	DOOR	
18 S . . .	SPELL	DOWNSTAIRS	
19 T -	WEST	UPSTAIRS	
20 U . . -	~~USE~~		
21 V . . . -	POSITION		
22 W . - -	DROP		
23 X - . . -	~~WEAR~~		
24 Y - . - -	~~TRAINER~~		
25 Z - - . .	~~SEARCH~~		

The Map

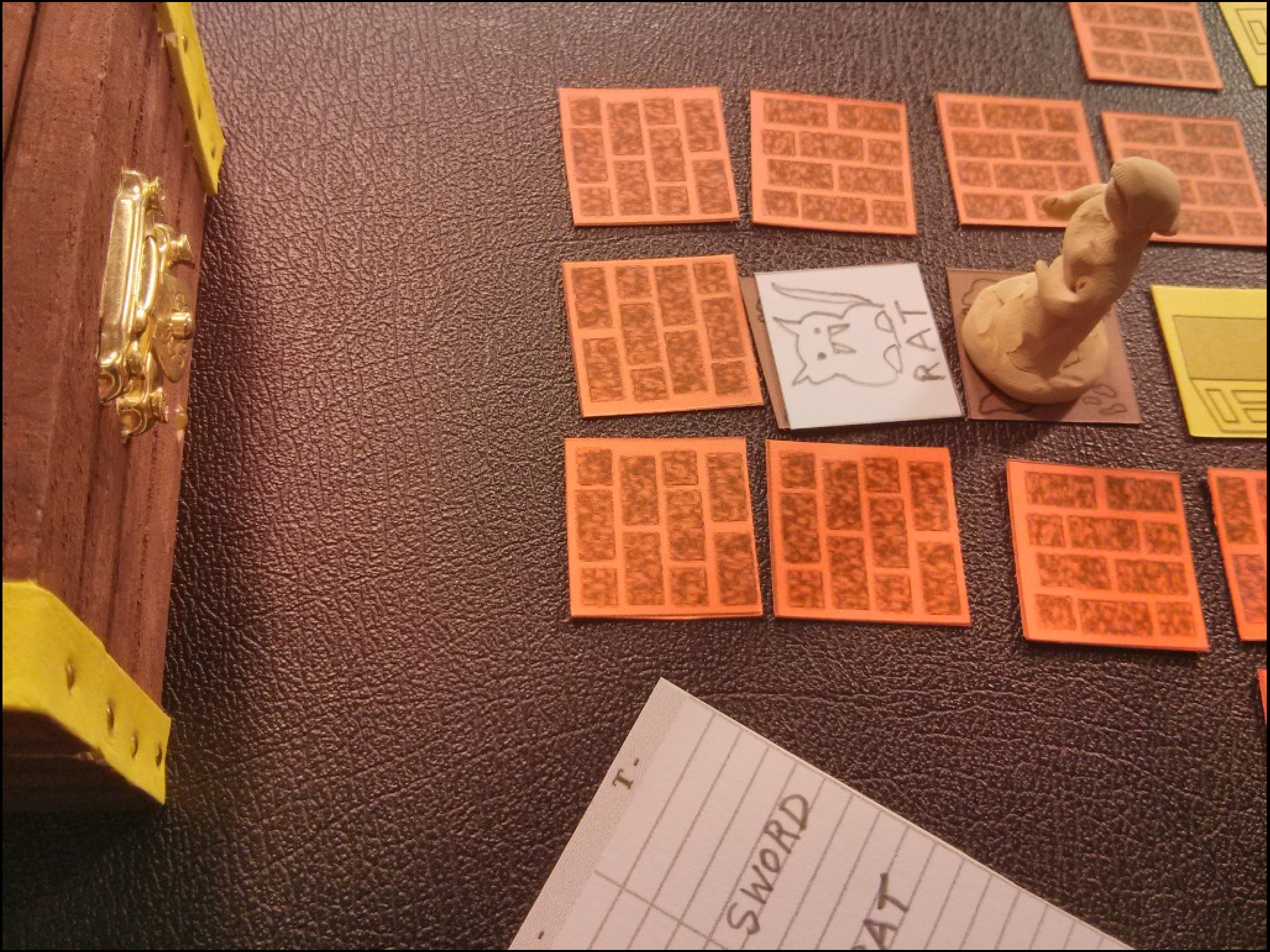
- 1 KB of ram is not enough to store a map, so procedurally generating a map cannot easily be done
- A 32x32 grid of 8-bit variables consumes 1024 bytes, all available RAM!
- Program space is important on AVRs since they are Harvard architecture devices. You can essentially move some data structures to flash (ROM). Modifying the flash while running (SPM) is possible, but it has to be done in pages. A good generator still requires a lot of space to work efficiently.
- It is ideal to generate the map on another computer and place it into program space during compilation (like 2-stage compiling)
- Levels must be large and shallow or small and deep due to low RAM/ROM
- Generator can easily use PRINTF and Unix redirection to save to a header .h file for compilation to AVR
- A gigantic monster/item pool can then be created on the generator machine and only a random subset is used for AVR compilation
- Halls one unit in size will save space (even rooms can be one unit)




YOU SEE
YOU FIND
YOU KILLED
YOU HEAR
YOU FEEL
SWORD
RAT



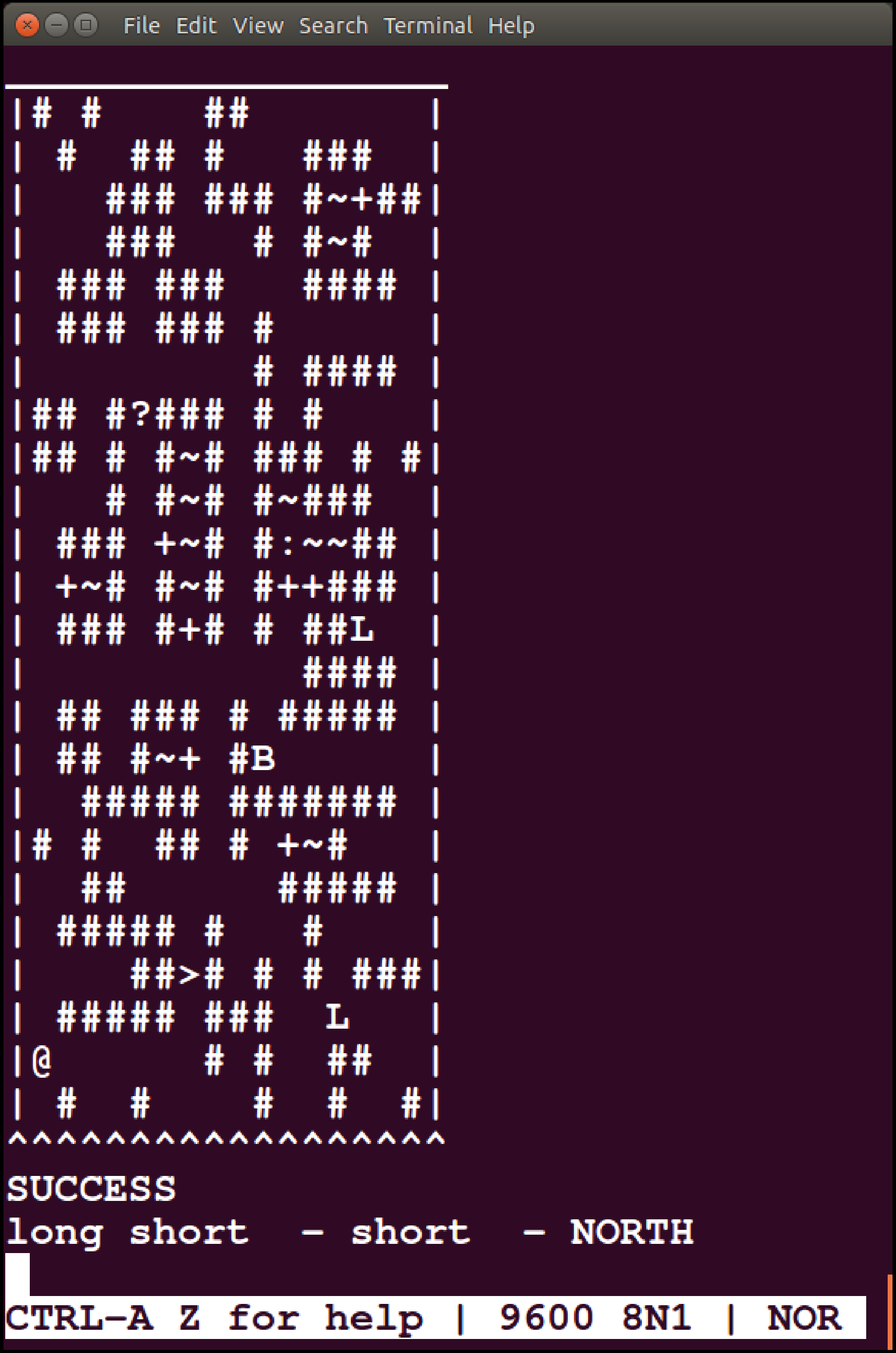
E.	A. -	T. -
SOUTH		
~~BUILD FIRE~~	~~YOU SEE~~	
CLIMB	YOU FIND	
~~DRINK EAT~~	YOU KILLED	
EAST	~~YOU HEAR~~	
LINDMODE	YOU FEEL	
T	YOU	SWORD
	TAKEN	
	DROPPED	
NTORY	MISS	
		RAT
	FLOOR	
	WALL	
	DOOR FRAME	
SE	WOODEN FLOOR	
	FLOOR HOLE	
	CEILING HOLE	
	DOOR	
	OWNSTAIRS	
	PSTAIRS	



RAT
T.
SWORD
RAT

AVR as UART Server

- The hardware UART using the internal oscillator works well for a small roguelike
- 9600 baud is fast enough to display small map
- ANSI escape codes can be used for more speed and control
- <esc>[2J will clear the screen between redraws
- Avoid using PRINTF on AVR to save space. Create PUT, PUTS, GET, GETS routines to talk to the UART registers directly



```
__________________
|# #    ##       |
| #  ## #   ###  |
|   ### ### #~+##|
|   ###   # #~#  |
| ### ###   #### |
| ### ### #      |
|         # #### |
|## #?### # #    |
|## # #~# ### # #|
|   # #~# #~###  |
| ### +~# #:~~## |
| +~# #~# #++### |
| ### #+# # ##L  |
|           #### |
| ## ### # ##### |
| ## #~+ #B      |
|  ##### ####### |
|# #  ## # +~#   |
|  ##      ##### |
| ##### #   #    |
|    ##># # # ###|
| ##### ###  L   |
|@      # #  ##  |
| #  #    #  #  #|
^^^^^^^^^^^^^^^^^^

SUCCESS
long short  - short  - NORTH

CTRL-A Z for help | 9600 8N1 | NOR
```

Leveraging CPU intelligence

- Circuit design can be simplified
- Can flip port states on the fly (input, output, tri-state, pull-ups)
- Logical tests for switch states, active scanning
- Use wake-up keys for power-up (interrupt)
- Blink mutually exclusive LED circuits faster than persistence of vision to simulate simultaneity
- De-bounce unreliable switches

Proof-of-Concept

- I built a proof-of-concept demo of a core game engine only; will add balanced gameplay later. It uses around 11K, leaving only 5K left.
- Made use of C struct bitfields to save space
- Stored items/actors in lists (arrays of structs), one with dynamic data (in RAM), one with static data (in Program Space)
- Map levels are stored as half-byte (nibble) arrays
- Used ragged 1-dimensional arrays to save space (are contiguous)
- Created message queue and repeat event flagging to limit messages
- Created Morse decoder, used lookup tables to reference words
- Created Success/Fail responses to limit messages
- Tried to make it both blind and deaf playable (LED + piezo)

Remember, anything is possible...

greatfractal.com/TinyRoomTinyWorld.html