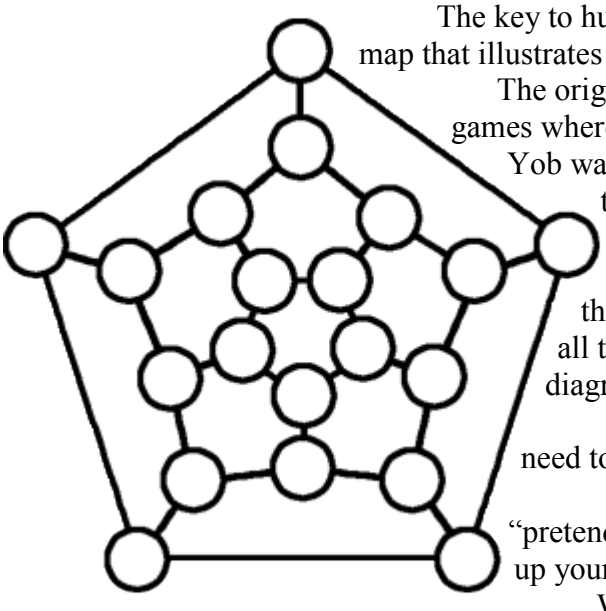
Hunt the Wumpus



The key to hunting the wumpus is the diagram to the left. It is the unfinished map that illustrates the interconnectedness of the tunnels in the game.

The original Hunt the Wumpus game was written as a reaction to all the games where you hunted some imaginary creature on a grid of points. Gregory Yob wanted to design a game with a slightly more dynamic map. He chose the edges and corner of a dodecahedron, a three-dimensional shape like a 10 sided dice. Since the map is designed in 3D you can imagine the caves are surrounding the center of some planet. But if that's too much the diagram to the left is a flattened dodecahedron, all the same connections but laid out in 2D. The first step is to take that diagram and number the caves correctly.

The game itself has instructions that explain everything else you need to know.

So, as the Eric S. Raymond said in the comments for this game, "pretend for a little while that your workstation is an ASR-33 and limber up your fingers for a trip to nostalgia-land…"

Wumpus.C was written by Eric S. Raymond, translated from a BASIC game by Gregory Yob.

WUMPUS.C | You will need: a C/C++ complier .

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <ctype.h>
static int path[5];
static int j, k, arrows, scratchloc;
static char inp[BUFSIZ];        /* common input buffer */
#define YOU    0
#define WUMPUS 1
#define PIT1   2
#define PIT2   3
#define BATS1  4
#define BATS2  5
#define LOCS   6
static int loc[LOCS], save[LOCS];    /* locations */
#define NOT    0
#define WIN    1
#define LOSE    -1
static int finished;
static int cave[20][3] = {
    {1,4,7},{0,2,9},{1,3,11},{2,4,13},{0,3,5},{4,6,14},{5,7,16},
    {0,6,8},{7,9,17},{1,8,10},{9,11,18},{2,10,12},{11,13,19},{3,12,14},
    {5,13,15},{14,16,19},{6,15,17},{8,16,18},{10,17,19},{12,15,18},
};
#define FNA() (rand() % 20)
#define FNB() (rand() % 3)
#define FNC() (rand() % 4)
int getnum(char *prompt) {
    (void) printf("%s\n?", prompt);
    (void) fgets(inp, sizeof(inp), stdin);
    return(atoi(inp));
}
int getlet(char *prompt) {
    (void) printf("%s\n?", prompt);
    (void) fgets(inp, sizeof(inp), stdin);
    return(toupper(inp[0]));
}
void print_instructions() {
    char ebuf[BUFSIZ];

    puts("\tWelcome to 'Hunt the Wumpus'\n\n"
    "  The Wumpus lives in a cave of 20 rooms. Each room has 3 tunnels\n"
    "leading to other rooms. (look at a dodecahedron to see how this works.\n"
    "If you don't know what a dodecahedron is, ask someone)\n");
    puts("     HAZARDS:\n"
    " Bottomless Pits - Two rooms have bottomless pits in them. If you go\n"
    "     there, you fall into the pit (& lose!)\n"
    " Super Bats - Two other rooms have super bats. If you go there a bat\n"
    "     grabs you and takes you to some other room at random. (Which may\n"
    "     be troublesome)\n\n"
    "     THE WUMPUS:\n"
    " The wumpus is not bothered by hazards (he has sucker feet and is too\n"
    " big for a bat to lift). Usually he is asleep. Two things wake him up:\n"
    " you shooting an arrow or you entering his room. If the wumpus wakes
```

Listing continued next column...

WUMPUS.C | Listing continued from previous column...

```
he\n"
    " moves (P=.75) one room or stays still (P=.25). After that, if he is\n"
    " where you are he eats you up and you lose!\n");
    (void) getlet("Type an E them RETURN ");
    puts("     YOU:\n"
    " Each turn you may move or shoot a crooked arrow\n"
    "   Moving: You can move one room (thru one tunnel)\n"
    "   Arrows: You have 5 arrows. You lose when you run out. Each arrow
can\n"
    "           go from 1 to 5 rooms. you aim by telling the computer the\n"
    "           room #s you want the arrow to go to. If the arrow can't go\n"
    "           that way (if no tunnel) it moves at random to the next
room.\n"
    "           If the arrow hits the wumpus, you win.\n"
    "           If the arrow hits you, you lose.\n\n"
    "     WARNINGS:\n"
    "When you are one room away from a wumpus or hazard the computer says:\n"
    " Wumpus:  'I smell a wumpus'\n"
    " BAT   :  'Bats nearby'\n"
    " PIT   :  'I feel a draft'\n");
    (void) getlet("Type an E them RETURN ");
}
void check_hazards() {
    (void) puts("");
    for (k = 0; k < 3; k++) {
        int room = cave[loc[YOU]][k];

        if (room == loc[WUMPUS])
            (void) puts("I smell a wumpus!");
        else if (room == loc[PIT1] || room == loc[PIT2])
            (void) puts("I feel a draft");
        else if (room == loc[BATS1] || room == loc[BATS2])
            (void) puts("Bats nearby!");
    }
    (void) printf("You are in room %d\n", loc[YOU]+1);
    (void) printf("Tunnels lead to %d %d %d\n\n",
                cave[loc[YOU]][0]+1, cave[loc[YOU]][1]+1, cave[loc[YOU]][2]
+1);
}
int move_or_shoot() {
    int c;
badin:
    c = getlet("Shoot or Move (S-M)");
    if (c == 'S')
        return(1);
    else if (c == 'M')
        return(0);
    else
        goto badin;
}
void shoot() {
    extern void check_shot(), move_wumpus();
    int    j9;
```

Listing continued on page 2...

WUMPUS.C | Listing continued from page 1...

```
    finished = NOT;
badrange:
    j9 = getnum("No. of rooms (1-5)");
    if (j9 < 1 || j9 > 5)
        goto badrange;
    for (k = 0; k < j9; k++) {
        path[k] = getnum("Room #") - 1;
        if (k <= 1)
            continue;
        if (path[k] != path[k - 2])
            continue;
        (void) puts("Arrows aren't that crooked - Try another room");
        k--;
    }
    scratchloc = loc[YOU];
    for (k = 0; k < j9; k++) {
        int k1;
        for (k1 = 0; k1 < 3; k1++) {
            if (cave[scratchloc][k1] == path[k]) {
                scratchloc = path[k];
                check_shot();
                if (finished != NOT)
                    return;
            }
        }
        scratchloc = cave[scratchloc][FNB()];
        check_shot();
    }
ammo:
    if (finished == NOT) {
        (void) puts("Missed");
        scratchloc = loc[YOU];
        move_wumpus();
        if (--arrows <= 0)
            finished = LOSE;
    }
}
void check_shot() {
    if (scratchloc == loc[WUMPUS]) {
        (void) puts("AHA! You got the wumpus!");
        finished = WIN;
    } else if (scratchloc == loc[YOU]) {
        (void) puts("OUCH! Arrow got you!");
        finished = LOSE;
    }
}
void move_wumpus() {
    k = FNC();
    if (k < 3)
        loc[WUMPUS] = cave[loc[WUMPUS]][k];
    if (loc[WUMPUS] != loc[YOU])
        return;
    (void) puts("Tsk tsk tsk - Wumpus got you!");
    finished = LOSE;
}
void move() {
    finished = NOT;
badmove:
    scratchloc = getnum("Where to");
    if (scratchloc < 1 || scratchloc > 20)
        goto badmove;
    scratchloc--;
    for (k = 0; k < 3; k++) {
        if (cave[loc[YOU]][k] == scratchloc)
            goto goodmove;
    }
    if (scratchloc != loc[YOU]) {
```

Listing continued next column...

WUMPUS.C | Listing continued from previous column...

```
        (void) puts("Not possible -");
        goto badmove;
    }
goodmove:
    loc[YOU] = scratchloc;
    if (scratchloc == loc[WUMPUS]) {
        (void) puts("... OOPS! Bumped a wumpus!");
        move_wumpus();
    } else if (scratchloc == loc[PIT1] || scratchloc == loc[PIT2]) {
        (void) puts("YYYYIIIIEEEE . . . Fell in pit");
        finished = LOSE;
    } else if (scratchloc == loc[BATS1] || scratchloc == loc[BATS2]) {
        (void) puts("ZAP--SUPER BAT SNATCH! Elsewhereville for you!");
        scratchloc = loc[YOU] = FNA();
        goto goodmove;
    }
}
main(int argc, char *argv) {
    int    c;

    if (argc >= 2 && strcmp(&argv[1], "-s") == 0)
        srand(atoi(&argv[2]));
    else
        srand((int)time((long *) 0));
    c = getlet("Instructions (Y-N)");
    if (c == 'Y')
        print_instructions();
badlocs:
    for (j = 0; j < LOCS; j++)
        loc[j] = save[j] = FNA();
    for (j = 0; j < LOCS; j++)
        for (k = 0; k < LOCS; k++)
            if (j == k)
                continue;
            else if (loc[j] == loc[k])
                goto badlocs;
newgame:
    arrows = 5;
    scratchloc = loc[YOU];
    (void) puts("HUNT THE WUMPUS");
#ifdef DEBUG
    (void) printf("Wumpus is at %d, pits at %d & %d, bats at %d & %d\n",
                  loc[WUMPUS]+1,
                  loc[PIT1]+1, loc[PIT2]+1,
                  loc[BATS1]+1, loc[BATS2]+1);
#endif
nextmove:
    check_hazards();
    if (move_or_shoot()) {
        shoot();
        if (finished == NOT)
            goto nextmove;
    } else {
        move();
        if (finished == NOT)
            goto nextmove;
    }
    if (finished == LOSE) {
        (void) puts("HA HA HA - You lose!");
    } else {
        (void) puts("Hee hee hee - The wumpus'll get you next time!!");
    }
    for (j = YOU; j < LOCS; j++)
        loc[j] = save[j];
    c = getlet("Same setup (Y-N)");
    if (c != 'Y')
        goto badlocs;
    else
        goto newgame;
}
```

Author's Notes:

WUMPUS.C — a faithful translation of the classic "Hunt The Wumpus" game. This was the state of the art 20 years ago, in 1972. We've come a long way, baby. The BASIC source is that posted by Magnus Olsson in USENET article <9207071854.AA21847@thep.lu.se>he wrote:

Below is the source code for _one_ (rather simple) Wumpus version, which I found in the PC-BLUE collection on SIMTEL20. I believe this is pretty much the same version that was published in David Ahl's "101Basic Computer Games" (or, possibly, in the sequel). (Editor's Note: it was the sequel, 'More BASIC Computer Games' edited by David H. Ahl © 1979)

I have staunchly resisted the temptation to "improve" this game. It is functionally identical to the BASIC version.... I fixed some typos in the help text.

Language hackers may be interested to know that he most difficult thing about the translation was tracking the details required to translate from 1-origin to 0-origin array indexing.

The only enhancement is a an -s command-line switch for setting the random number seed.