Code Block Ray Caster

The code may not be pretty, but for a program so small this program does something amazing. Originally written to fit use no more than 80x24 characters this program creates a 3D view of a maze for you to navigate from a rats eye view. The listing here has been reformatted to make it slightly more readable. It uses a technique called ray casting which was employed by early 3D video games like Wolfenstein 3D and Doom to essentially extrude a 2D map and fake a 3D appearance.

In Codeblock Raycaster you must navigate a maze seeking for the fabled red block. When you find it a new red block will appear. Find 5 red blocks and you win the game. Use the arrow keys to move. Press the escape key to exit.

This program is an excellent starting point, but it could use a few things. You could easily improve the game with a randomly generated map. If you were a bit more adept you could figure out a way to change the goals from red squares to gems or treasure chests.

Codeblock Raycaster was written by Terry Cavanagh.

```
/* codeblockraycaster.c linsting begins: */

#include <allegro.h> /* 80x24 Codeblock:  Simple Raycaster Maze  1st Jul '08 */
#include <math.h>    /* Terry Cavanagh   |   http://www.distractionware.com  */
BITMAP *b;
int xp,yp,n,m;
int map[15][15]={
{1,1,1,1,1,1,1,1,1,1,1,1,1,1,1},
{1,0,0,0,0,1,0,1,0,1,0,0,0,0,1},
{1,0,2,1,0,0,0,0,0,0,0,0,2,0,1},
{1,0,1,1,1,1,1,1,1,1,1,0,0,0,1},
{1,0,0,1,0,0,0,0,0,0,0,0,0,0,1},
{1,1,0,1,0,2,2,0,2,2,0,1,1,0,1},
{1,0,0,1,0,2,0,0,0,2,0,1,0,0,1},
{1,0,1,1,0,0,0,1,0,0,0,1,0,1,1},
{1,0,0,1,0,2,0,0,0,2,0,1,0,0,1},
{1,1,0,1,0,2,2,0,2,2,0,1,1,0,1},
{1,0,0,0,0,0,0,0,0,0,0,0,1,0,1},
{1,0,0,0,0,1,1,1,1,1,1,1,1,0,1},
{1,0,2,0,0,0,0,0,0,0,0,1,2,0,1},
{1,0,0,0,0,1,0,1,0,1,0,0,0,0,1},
{1,1,1,1,1,1,1,1,1,1,1,1,1,1,1}};

int z(int q,int d) {
    if (q==3)return makecol(255,0,0);
    d=128-d;
    if (d<0)d=0;
    if (q<2)return makecol(d,d,d);
    if (q==2)return makecol(d,d,0);
}

void r(int x,int y,int s,int c) {
    rectfill(b,x,y,x+10,y+s,c);
}

void pl() {
    m=1;
    while (m>0) {
        xp=rand()%15;
        yp=rand()%15;
        m=map[xp][yp];
    }
    map[xp]
    [yp]=3;
    n++;
}

void re() {
    map[xp][yp]=m;
    pl();
}

int main() {
    allegro_init();
    srand(time(0));
    install_keyboard();
    double u=0,x=100,y=100,d=0,c,t,s,q,f,g;
    int w=640,h=480;

    set_color_depth(32);
    set_gfx_mode(2,w,h,0,0);
    b=create_bitmap(w,h);
    n=-1;
```

Listing continued on next page...

Listing continued from previous page

```
    pl();
    while (!key[KEY_ESC]&&u==0) {
        acquire_bitmap(b);
        clear_bitmap(b);
        rectfill(b,0,240,640,480,makecol(32,32,32));
        int i; for (i=0;i<64;i++) {
            c=0;
            t=x;
            s=y;
            while (map[(int)(t/100)][(int)(s/100)]==0 && c<800) {
                c++;
                q=d+(i*0.0174)-0.5585;
                t+=sin(q);
                s+=cos(q);
            }
            c=(10000/c)/cos(i*0.014);
            r(i*10,240-c,2*c,z(map[(int)(t/100)][(int)(s/100)],2000/c))
            ;
        }
        for (i=0;i<n;i++) {
            r(5+(i*15),465,10,makecol(255,0,0));
        }
        blit(b,screen,0,0,0
             ,0,w,h);
        release_bitmap(b);
        vsync();
        f=x;
        g=y;
        if (key[84]) {
            x+=sin(d)*10;
            y+=cos(d)*10;
        }
        if (key[82])d-=0.125;
        if (key[85]) {
            x-=sin(d)*10;
            y-=cos(d)*10;
        }
        if (key[83])d+=0.125;
        c=map[(int)(x/100)][(int)(y/100)];
        if (c!=0) {
            if (c==3)re();
            x=f;
            y=g;
        }
        if (n>=5)u=1;
    }
    return 0;
}
END_OF_MAIN()
```