## Dodge

Survive as long as you can. Touching the circles will kill you. Touching the swelling cross will kill you. You are only given a moment's warning before the swelling cross becomes deadly. Keeping moving is the only way to stay alive.

You control the hollow circle with the mouse. When you lose you can start a new game with the right mouse button.

The game is simple, and so are the controls, but the satisfaction of survival is

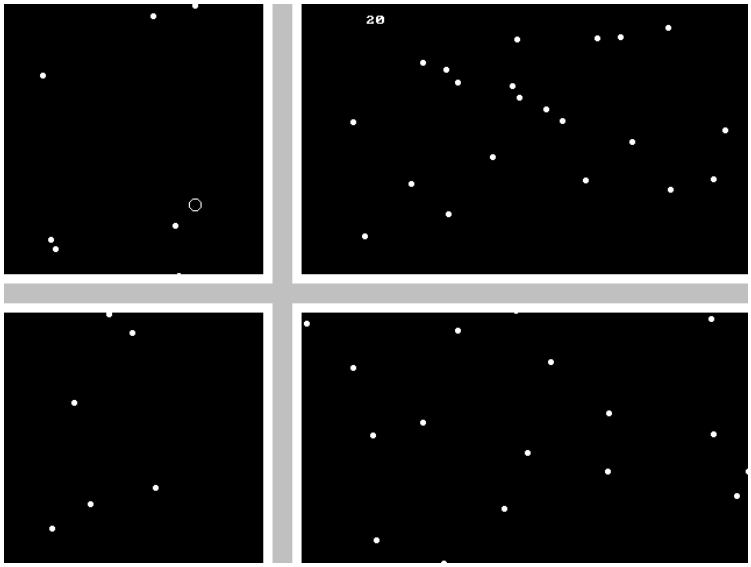
Dodge is by Jacob Charles, AKA Rio Kikaru , submitted as an entry for MinorHack at September 27th at 7:00 pm UTC.

// dodge.cpp listing begins:

```
#include <allegro.h>

#define MODE GFX_AUTODETECT_WINDOWED
#define WIDTH 640
#define HEIGHT 480
#define COLOR_DEPTH 32

#define BLACK makecol(0, 0, 0)
#define SILVER makecol(192, 192, 192)
#define WHITE makecol(255, 255, 255)

#define LOOP_TIME 17

#define _ALLEGRO_NO_FIX_CLASS_

BITMAP *buffer;
long start_time;
long long timer = 0;
int loop_num = 0;
bool end_game = false;
bool game_over = false;

class bullet
{
public:
int x, y, vx, vy;
};

bool hit(int x, int y, int x2, int y2, int d)
{
    return ((abs(x-x2) < d)&&(abs(y-y2) < d));
}

//objects
bullet shot[1024];
int nshots = 0;
int ray_x, ray_y, ray_t = 50;

//player
int mx, my;

int score = 0;
int F_PLUS;

//update the timer
void update_timer(){timer++;}

//close button
void quit(){end_game = true;}

//main function
void main()
{
    //setup allegro
    allegro_init();
    install_mouse();
    install_timer();
    set_color_depth(COLOR_DEPTH);
    set_gfx_mode(MODE, WIDTH, HEIGHT, 0, 0);
    //setup the timer
    install_int(update_timer, 1);
```

// Listing continued on next page…

// Listing continued from previous page




```
    //seed random number
    srand(time(NULL));
    //set up the close button
    set_close_button_callback(quit);
    //initialize the buffer
    buffer = create_bitmap(WIDTH, HEIGHT);
    //main loop
    while (!end_game)
    {
        //log the start time
        start_time = timer;

        F_PLUS = (loop_num/50);
        if (F_PLUS > 120) F_PLUS = 120;

        mx = (mouse_x+mx*7)/8;
        my = (mouse_y+my*7)/8;

        if (mouse_b & 2) {nshots = 0; score = 0; ray_t = 50; game_over = false;}

        //fire a new shot
        if (loop_num % 10-F_PLUS/30 == 0)
        {
                if (rand()%2)
                {
                        shot[nshots].x = rand()%2*WIDTH;
                        shot[nshots].y = rand()%HEIGHT;
                        shot[nshots].vy = 0;
                        shot[nshots].vx = (WIDTH/2-shot[nshots].x)/(WIDTH/2);
                        nshots++;
                }
                else
                {
                        shot[nshots].y = rand()%2*HEIGHT;
                        shot[nshots].x = rand()%WIDTH;
                        shot[nshots].vx = 0;
                        shot[nshots].vy = (HEIGHT/2-shot[nshots].y)/(HEIGHT/2);
                        nshots++;
                }
        }
        //rays
        if ((loop_num % 300-F_PLUS == 0)&&(loop_num > 300))
        {
                ray_x = mx+rand()%11-5;
                ray_y = my+rand()%11-5;
                ray_t = -40;
        }

        if (loop_num % 2) ray_t++;

        //bullet move
        for (int i = 0; i < nshots; i++)
        {
                shot[i].x += shot[i].vx;
                shot[i].y += shot[i].vy;
                if ((shot[i].x > WIDTH)||(shot[i].y > HEIGHT)
                ||(shot[i].x < 0)||(shot[i].y < 0))
                  {nshots--; shot[i] = shot[nshots];}
                if (hit(mx, my, shot[i].x, shot[i].y, 5)){game_over = true;}
        }

        //death
        if ((abs(ray_t) < 20)&&(((mx < ray_x+(20-abs(ray_t)))&&(mx > ray_x-(20-
```

// Listing continued on next page…

// Listing continued from previous page

```
abs(ray_t))))
        ||((my < ray_y+(20-abs(ray_t)))&&(my > ray_y-(20-abs(ray_t))))))
{game_over = true;}

        ////drawing////

        //player
        if (!game_over)
        circle(buffer, mx, my, 5, WHITE);
        else
        circle(buffer, mx, my, 5, makecol(255, 0, 0));

        //bullets
        for (int i = 0; i < nshots; i++)
        {
                circlefill(buffer, shot[i].x, shot[i].y, 2, WHITE);
        }

        //beams
        if (ray_t <= -20)
        {
                hline(buffer, 0, ray_y, WIDTH, WHITE);
                vline(buffer, ray_x, 0, HEIGHT, WHITE);
        }
        if (abs(ray_t) < 20)
        {
                rectfill(buffer, 0, ray_y-(20-abs(ray_t)), WIDTH,
                  ray_y+(20-abs(ray_t)), WHITE);
                rectfill(buffer, ray_x-(20-abs(ray_t)), 0, ray_x+(20-abs(ray_t)),
                  HEIGHT, WHITE);
                rectfill(buffer, 0, ray_y-(20-abs(ray_t))/2, WIDTH,
                  ray_y+(20-abs(ray_t))/2, SILVER);
                rectfill(buffer, ray_x-(20-abs(ray_t))/2, 0,
                  ray_x+(20-abs(ray_t))/2, HEIGHT, SILVER);
        }
        //text
        if (!game_over)
        textprintf_centre_ex(buffer, font, WIDTH/2, 10, WHITE, BLACK, "%i"
          , score*LOOP_TIME/1000);
        else
        {
        textprintf_centre_ex(buffer, font, WIDTH/2, 10, makecol(255, 0, 0),
          BLACK, "%i", score*LOOP_TIME/1000);
        textprintf_centre_ex(buffer, font, WIDTH/2, HEIGHT/2-5, WHITE, BLACK,
          "Game Over");
        textprintf_centre_ex(buffer, font, WIDTH/2, HEIGHT/2+10, WHITE, BLACK,
          "Right-Click to restart");
        }
        //draw to the screen
        blit(buffer, screen, 0, 0, 0, 0, WIDTH, HEIGHT);
        //clear the buffer
        clear_to_color(buffer, BLACK);
        //wait
        while (timer < start_time + LOOP_TIME)
        {rest(1);}
        loop_num++;
        if (game_over == false) score++;
    }
    remove_int(update_timer);
    destroy_bitmap(buffer);
    return;
}
END_OF_MAIN()
```