Arrow Fighter

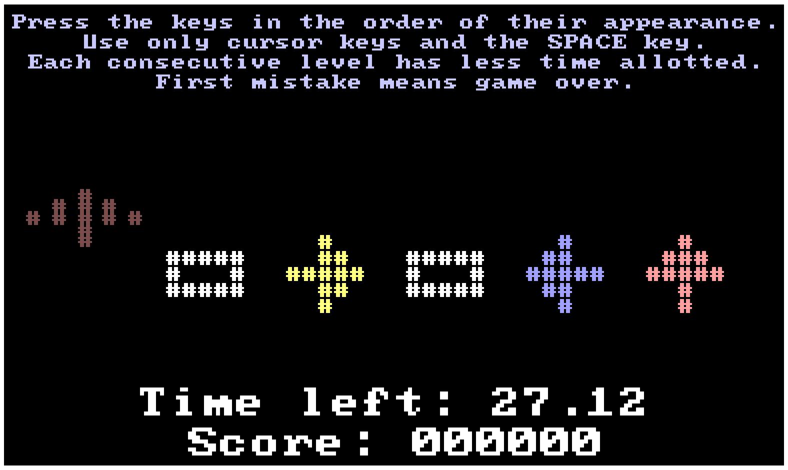
Speed and accuracy is the name of this game. Type in the sequence you see as fast as you can. Each level gives you less and less time. A simple premise, but what level can you make it to. The game keeps your high score so you can always challenge yourself.

Arrow Fighter was an entry in the March 17 2007 MinorHack challenge, meaning it had to be written (first draft) in an hour, be contained entirely within one source file, and the extra rule for that competition was: View The game must only use Allegro's text output functions with the default font for drawing. So while this program requires Allegro at first glance it may look like a Curses application, at least until it starts moving.

Arrow Fighter was written by Jakub Wasilewski

/* arrowfighter.c linsting begins: */

```
/* Minorhack entry */
/* by: Jakub Wasilewski */

#include <allegro.h>
#ifdef ALLEGRO_WINDOWS
#include <winalleg.h>
#include <windows.h>
#endif

#include <string>
#include <vector>
#include <cstdlib>
#include <ctime>
#include <cctype>
#include <cmath>

using namespace std;

volatile int ticks = 0;
void tick()
{
	ticks++;
}

/*********************************/
char *arrows[5][5] =
{
{"  #  ",
 " ### ",
 "#####",
 "  #  ",
 "  #  "},

{"  #  ",
 "  ## ",
 "#####",
 "  ## ",
 "  #  "},

{"  #  ",
 "  #  ",
 "#####",
 " ### ",
 "  #  "},

{"  #  ",
 " ##  ",
 "#####",
 " ##  ",
 "  #  "},

{"     ",
 "#####",
 "#   #",
 "#####",
 "     "}
};

/*********************************/
struct Score
{
```



```
Press the keys in the order of their appearance.
  Use only cursor keys and the SPACE key.
 Each consecutive level has less time allotted.
      First mistake means game over.

           Time left: 27.12
            Score: 000000
```

/* Listing continued on next page…*/

/* Listing continued from previous page */

```
        string name;
        int score, level;
};

/*********************************/
int keys[5] = {KEY_UP, KEY_RIGHT, KEY_DOWN, KEY_LEFT, KEY_SPACE};
double acol[5][3] = {{255.0, 160.0, 160.0}, {255.0, 255.0, 130.0}, {140.0, 255.0,
140.0}, {160.0, 160.0, 255.0}, {255.0, 255.0, 255.0}};
BITMAP *buf, *buf2;
vector<Score> hsc;

/*********************************/
struct Character
{
        double x, y;
        double vx, vy;
        char ch;

        Character(){};
        Character(double x, double y, char ch) : x(x), y(y), ch(ch) {vx=0.0; vy =
0.0;};
        void update()
        {
                x += vx;
                y += vy;
                vx *= 0.99;
                vy *= 0.99;
        }
        void draw(int color)
        {
                textprintf_ex(buf, font, (int)x, (int)y, color, -1, "%c", ch);
        }
};

struct Arrow
{
        int type;
        Character chars[25];
        bool done;
        double life;

        Arrow()
        {
        }

        Arrow(int type, int num, int outOf)
                : type(type)
        {
                int fullWidth = outOf * 60 - 10;
                int baseX = (320 - fullWidth / 2) + num * 60;
                int baseY = 220;

                for (int i = 0; i < 5; i++)
                        for (int j = 0; j < 5; j++)
                                chars[i * 5 + j] = Character(baseX + 8 * j, baseY +
8 * i, arrows[type][i][j]);

                done = false;
                life = 1.0;
        }

        void update()
        {
```

/* Listing continued on next page…*/

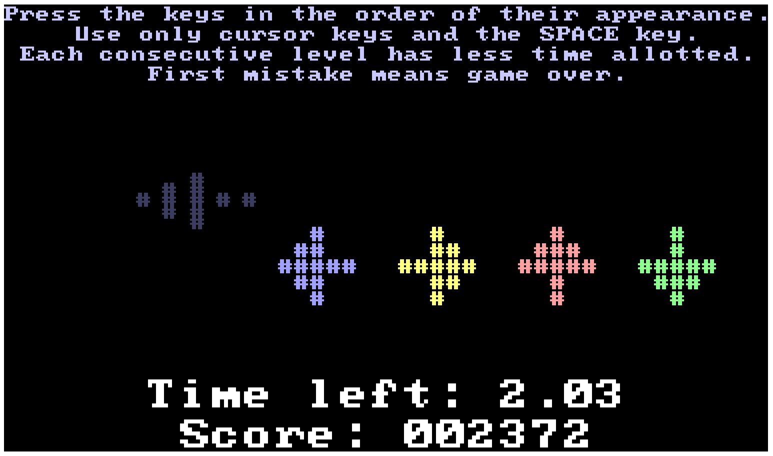
/* Listing continued from previous page. */

```
		if (done)
		{
			life -= 0.02;
			if (life <= 0.0)
				return;
		}

		for (int i = 0; i < 25; i++)
			chars[i].update();
	}

	void asplode()
	{
		done = true;
		for (int i = 0; i < 25; i++)
		{
			int bx = i % 5, by = i / 5;
			chars[i].vx = (bx - 2) * 0.2;
			chars[i].vy = -1.0 - (by * 0.1);
		}
	}

	void draw()
	{
		if (life <= 0.0) return;

		int col = makecol((int)(acol[type][0] * life),
			(int)(acol[type][1] * life),
			(int)(acol[type][2] * life));

		for (int i = 0; i < 25; i++)
			chars[i].draw(col);
	}
};

/*********************************/
bool end = false;
double timeLeft;
Arrow a[6];
int where;
double score;

/*********************************/
void init()
{
	allegro_init();
	set_color_depth(32);
	set_gfx_mode(GFX_AUTODETECT_WINDOWED, 640, 480, 0, 0);

	install_keyboard();
	install_timer();

	install_int_ex(tick, BPS_TO_TIMER(100));

	buf = create_bitmap(640, 480);
	buf2 = create_bitmap(640, 480);

	srand(time(NULL));
}

/*********************************/
char playerName[128];
void setPlayerName()
```

/* Listing continued on next page…*/

/* Listing continued from previous page */

```
{
#if defined(ALLEGRO_WINDOWS)
        DWORD len = 128;
        GetUserName(playerName, &len);
#elif defined(ALLEGRO_LINUX)
        strcpy(playerName, getenv("USER"));
#else
        strcpy(playerName, "Player");
#endif

        playerName[0] = toupper(playerName[0]);
        playerName[15] = '\0';
}

/********************************/
void newLevel(int no)
{
        for (int i = 0; i < 6; i++)
        {
                a[i] = Arrow(rand() % 5, i, 6);
        }

        if (no == 1)
                timeLeft = 30.0;
        else
                timeLeft = std::pow(0.89, no - 2) * 5.0;

        where = 0;
}

/********************************/
void loadHighscore()
{
        set_config_file("arrowfighter.hsc");
        char varName[512];
        const char *name;
        int score, level;

        for(int i = 0; i < 10; i++)
        {
                sprintf(varName, "name.%d", i);
                name = get_config_string("", varName, "Nobody");
                sprintf(varName, "score.%d", i);
                score = get_config_int("", varName, 0);
                sprintf(varName, "level.%d", i);
                level = get_config_int("", varName, 1);
                Score s = {name, score, level};
                hsc.insert(hsc.end(), s);
        }
}

/********************************/
void saveHighscore()
{
        set_config_file("arrowfighter.hsc");
        char varName[512];

        for(int i = 0; i < 10; i++)
        {
                sprintf(varName, "name.%d", i);
                set_config_string("", varName, hsc[i].name.c_str());
                sprintf(varName, "score.%d", i);
                set_config_int("", varName, hsc[i].score);
```



```
Press the keys in the order of their appearance.
   Use only cursor keys and the SPACE key.
 Each consecutive level has less time allotted.
        First mistake means game over.

          Time left: 2.03
           Score: 002372
```

/* Listing continued on next page…*/

/* Listing continued from previous page. */

```
		sprintf(varName, "level.%d", i);
		set_config_int("", varName, hsc[i].level);
	}

	flush_config_file();
}

/*********************************/
int main()
{
	init();
	loadHighscore();
	setPlayerName();

	int col;
	int k;

	do
	{

	clear_bitmap(screen);
	clear_bitmap(buf);
	textprintf_centre_ex(buf, font, 160, 8, makecol(255, 255, 100), -1, "**
ARROW FIGHTER **");
	textprintf_centre_ex(buf, font, 160, 50, makecol(255, 255, 120), -1, "%-
16s  %-2s  %-7s", "Name", "Lv", "Score");
	for (int i = 0; i < 10; i++)
		textprintf_centre_ex(buf, font, 160, 65 + i * 10, makecol(255 - i
* 20, 255 - i * 20, 255 - i * 4), -1, "%-16s  %2d  %07d", hsc[i].name.c_str(),
hsc[i].level, hsc[i].score);
	stretch_blit(buf, screen, 0, 0, 320, 240, 0, 30, 640, 480);
	textprintf_centre_ex(screen, font, 320, 400, makecol(100, 100, 255), -1,
"Press ENTER to begin, ESC to exit");

	do
	{
		k = readkey() >> 8;

	} while ((k != KEY_ENTER) && (k != KEY_ESC));
	if (k == KEY_ESC) break;

	int levelNo = 1;
	score = 0;
	newLevel(1);
	ticks = 0;

	while ((timeLeft > 0.0) and (!key[KEY_ESC]))
	{
		while (ticks > 0)
		{
			for (int i = 0; i < 6; i++)
				a[i].update();

			if (keypressed())
			{
				int pressed = readkey() >> 8;
				if (pressed == keys[a[where].type])
				{
					a[where].asplode();
					where++;
				}
				else if (pressed != KEY_ENTER)
				{
```

/* Listing continued on next page…*/

/* Listing continued from previous page */

```
                                        timeLeft = 0.0;
                                }
                        }

                        if (where < 6)
                                timeLeft -= 0.01;
                        else
                                score += ((levelNo - 1) * 3.85) * pow(1.05 +
(levelNo - 2) * 0.16, 1.0 + timeLeft * 2.0);


                        if (a[5].life <= 0.0)
                        {
                                newLevel(++levelNo);
                        }

                        ticks--;
                }

                clear_bitmap(buf);
                clear_bitmap(buf2);
                for (int i = 0; i < 6; i++)
                        a[i].draw();
                textprintf_centre_ex(buf, font, 320, 110, makecol(200, 200, 255),
-1, "Press the keys in the order of their appearance.");
                textprintf_centre_ex(buf, font, 320, 120, makecol(200, 200, 255),
-1, "Use only cursor keys and the SPACE key.");
                textprintf_centre_ex(buf, font, 320, 130, makecol(200, 200, 255),
-1, "Each consecutive level has less time allotted.");
                textprintf_centre_ex(buf, font, 320, 140, makecol(200, 200, 255),
-1, "First mistake means game over.");

                if (timeLeft > 2.0)
                        col = makecol(255, 255, 255);
                else if (timeLeft > 1.0)
                        col = makecol(255, (int)(255.0 * (timeLeft - 1.0)),(int)
(255.0 * (timeLeft - 1.0)));
                else
                        col = makecol(255, (int)(255.0 * (timeLeft - 0.0)),(int)
(255.0 * (timeLeft - 0.0)));

                textprintf_centre_ex(buf2, font, 320, 300, col, -1, "Time left: %
0.2lf", timeLeft);
                textprintf_centre_ex(buf2, font, 320, 310, makecol(255, 255, 255),
-1, "Score: %06d", (int)score);
                stretch_blit(buf2, buf, 240, 298, 160, 24, 160, 292, 320, 48);
                blit(buf, screen, 0, 0, 0, 0, 640, 480);
        }

        int iScore = (int)score;
        vector<Score>::iterator it;
        Score s = {playerName, iScore, levelNo};
        for (it = hsc.begin();;it++)
                if (it == hsc.end())
                {
                        hsc.insert(it, s);
                        break;
                }
                else if ((*it).score < iScore)
                {
                        hsc.insert(it, s);
                        break;
                }
```

/* Listing continued on next page…*/

/* Listing continued from previous page. */

```
        clear_bitmap(screen);
        textprintf_centre_ex(screen, font, 320, 240, makecol(255, 64, 64), -1,
"*** GAME OVER ***");
        textprintf_centre_ex(screen, font, 320, 250, makecol(255, 128, 64), -1,
"Level achieved: %d", levelNo);
        textprintf_centre_ex(screen, font, 320, 260, makecol(255, 196, 64), -1,
"Score accumulated: %d", (int)score);
        textprintf_centre_ex(screen, font, 320, 270, makecol(255, 255, 64), -1,
"Press ENTER to continue");

        while (keypressed()) readkey();

        do
        {
                k = readkey() >> 8;
        } while (k != KEY_ENTER);

        } while (true);

        saveHighscore();
        return 0;
}
END_OF_MAIN();
```