Poison Water

This is a take on the classic shell game, only this time with glasses of water. Presumably one is water, the other 4 are poison, tho which is which changes after you win a round.

Either way the game is simple, there are 5 animated glasses of clear liquid. One will flash and then before your eyes they'll shuffle. Keep your eye on the one that flashed because when you get the chance pick that one to win and go on to the next round. How many rounds until it's too fast for you?

Poison Water is written by Jakub Wasilewski for the MinorHack on April 14, 2007.

// poisonwater.cpp listing begins:

```
#include <allegro.h>
#include <cmath>
#include <cstdlib>
#include <ctime>

using namespace std;
/****************************************/
BITMAP *arrow;
BITMAP *buf;
BITMAP *glass;

volatile static int timer = 0;
void tick() { timer++; }
/****************************************/
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);
  glass = create_bitmap(50, 100);
  clear_to_color(glass, makecol(255, 0, 255));
  ellipse(glass, 25, 85, 16, 10, makecol(255, 255, 255));
  rectfill(glass, 0, 0, 50, 85, makecol(255, 0, 255));
  ellipse(glass, 25, 15, 24, 15, makecol(255, 255, 255));
  line(glass, 9, 85, 0, 15, makecol(255, 255, 255));
  line(glass, 41, 85, 49, 15, makecol(255, 255, 255));
  for (double f = 0.99; f >= 0.0; f -= 0.05) {
    int y = (int)(35.0 + f * 50.0);
    int r1 = (int)(22.0 - f * 6.0);
    int r2 = (int)(13.0 - f * 3.0);
    ellipsefill(glass, 25, y, r1, r2,
      makecol(0, 120 - (int)(f * 120.0), 255 - (int)(f * 30.0)));
  }
  ellipsefill(glass, 25, 35, 21, 12, makecol(0, 130, 255));
  ellipse(glass, 25, 15, 24, 15, makecol(255, 255, 255));
  line(glass, 9, 85, 0, 15, makecol(255, 255, 255));
  line(glass, 41, 85, 49, 15, makecol(255, 255, 255));
  arrow = create_bitmap(30, 30);
  clear_to_color(arrow, makecol(255, 0, 255));
  rectfill(arrow, 10, 10, 20, 30, makecol(255, 255, 0));
  triangle(arrow, 15, 0, 29, 15, 0, 15, makecol(255, 255, 0));

  srand(time(NULL));
}
/****************************************/
const int GLASS_COUNT = 5;
enum {TRANS_DOWN = 1, TRANS_UP = -1, TRANS_NONE = 0};

class Glass {
  struct Bubble {
    double x, y, life;
  };

public:
  double transF;
```

// Listing continued on next page…

// Listing continued from previous page

```
  int prevPos, nowPos;
  int trans;
  bool rightOne;
  Bubble bubbles[5];

  Glass() {};
  Glass(int position) : nowPos(position), prevPos(position), trans(TRANS_NONE)
    , transF(0.0), rightOne(false) {
    for (int i = 0; i < 5; i++)
      newBubble(i);
  };

  void update(double dt) {
    if (trans != TRANS_NONE) {
      transF += dt;
      if (transF > 1.00) {
        trans = TRANS_NONE;
        transF = 0.0;
        prevPos = nowPos;
      }
    }
    for (int i = 0; i < 5; i++) {
      bubbles[i].y -= dt * 20.0;
      bubbles[i].life -= dt;
      if (bubbles[i].life < 0.0)
        newBubble(i);
    }
  };

  void newBubble(int i) {
    bubbles[i].x = 16.0 + 0.18 * (rand() % 100);
    bubbles[i].y = 90.0 - 0.20 * (rand() % 100);
    bubbles[i].life = 1.2 - 0.005 * (rand() % 100);
  }
```


```
  void render(BITMAP *bmp) {
    double x = 340.0 - 40.0 * GLASS_COUNT;
    double xAdd = nowPos * 80.0 * transF + prevPos * 80.0 * (1.0 - transF);
    double y = 150.0 + trans * sin(transF * M_PI) * 20.0;
    set_trans_blender(255, 255, 255, (int)(y * 3.0 - 270.0));
    draw_trans_sprite(bmp, glass, (int)(x + xAdd), (int)y);
    drawing_mode(DRAW_MODE_TRANS, NULL, 0, 0);
    for (int i = 0; i < 5; i++) {
      set_trans_blender(255, 255, 255, (int)(64.0 * bubbles[i].life));
      circle(bmp, (int)(x+xAdd+bubbles[i].x), (int)(y + bubbles[i].y), 2,
        makecol(255, 255, 255));
    }
    drawing_mode(DRAW_MODE_SOLID, NULL, 0, 0);
  };
};
/*****************************************/
Glass glasses[5];

enum {GAME_SHOW, GAME_SWITCH, GAME_CHOOSE, WRONG};

int main() {
  init();

  int kp;
  do {
    for (int i = 0; i < 5; i++)  glasses[i] = Glass(i);
    glasses[rand() % 5].rightOne = true;
```

// Listing continued on next page…

// Listing continued from previous page

```
    timer = 0;
    bool end = false;
    int timing = 0;
    int state = GAME_SHOW;
    int level = 1, switches;
    int arrowPos = 2;

    while (!end) {
      while (timer > 0) {
        if (key[KEY_ESC]) {
          end = true;
          break;
        }
        for (int i = 0; i < 5; i++)
          glasses[i].update(0.015 + pow((double)level, 0.8) * 0.004);
        timing++;
        if ((timing == 50) && state == GAME_SHOW) {
          state = GAME_SWITCH;
          timing = 0;
          switches = 0;
        }
        if (state == GAME_SWITCH) {
          bool sw = true;
          for (int i = 0; i < 5; i++)
            if (glasses[i].trans != TRANS_NONE) sw = false;
          if (sw) {
            int glass1 = rand() % 5;
            if (rand() % 2) {
              for (int i = 0; i < 4; i++)
                if (glasses[i].rightOne) glass1 = i;
            }
            int glass2;
            do { glass2 = rand() % 5; } while (glass1 == glass2);
            if (rand() % 2) {
              int t = glass2;
              glass2 = glass1;
              glass1 = t;
            }
            glasses[glass1].nowPos = glass2;
            glasses[glass2].nowPos = glass1;
            glasses[glass1].trans = TRANS_DOWN;
            glasses[glass2].trans = TRANS_UP;
            Glass g = glasses[glass1];
            glasses[glass1] = glasses[glass2];
            glasses[glass2] = g;
            switches++;
            if (switches >= 5 + level * 2) { state = GAME_CHOOSE; arrowPos = 2; }
          }
        }
        if (state == GAME_CHOOSE)
          if (keypressed()) {
            int k = readkey() >> 8;
            if ((k == KEY_LEFT) && (arrowPos > 0)) arrowPos--;
            if ((k == KEY_RIGHT) && (arrowPos < 4)) arrowPos++;
            if (k == KEY_ENTER || k == KEY_SPACE) {
              if (glasses[arrowPos].rightOne) {
                for (int i = 0; i < 5; i++) glasses[i] = Glass(i);
                glasses[rand() % 5].rightOne = true;
                arrowPos = 2;
                level++;
                state = GAME_SHOW;
                timing = 0;
              } else { end = true; state = WRONG; }
```

// Listing continued on next page…

// Listing continued from previous page

```
            }
          }
        timer--;
      }
      clear_bitmap(buf);
      for (int i = 0; i < 5; i++)
        if (((!glasses[i].rightOne) || (state != GAME_SHOW)
          || (timing % 20 > 10)) && glasses[i].trans == TRANS_UP)
          glasses[i].render(buf);
      for (int i = 0; i < 5; i++)
        if (((!glasses[i].rightOne) || (state != GAME_SHOW)
          || (timing % 20 > 10)) && glasses[i].trans == TRANS_NONE)
          glasses[i].render(buf);
      for (int i = 0; i < 5; i++)
        if (((!glasses[i].rightOne) || (state != GAME_SHOW)
          || (timing % 20 > 10)) && glasses[i].trans == TRANS_DOWN)
          glasses[i].render(buf);
      if (state == GAME_CHOOSE)
        draw_sprite(buf, arrow, 150 + arrowPos * 80, 280);
      textprintf_centre_ex(buf, font, 320, 20, makecol(255, 170, 0), -1,
        "ALWAYS PICK THE FLASHING GLASS!");
      textprintf_centre_ex(buf, font, 320, 360, makecol(180, 180, 255), -1,
        "LEVEL: %d", level);
      if (state == WRONG)
        textprintf_centre_ex(buf, font, 320, 370, makecol(255, 0, 0), -1,
          "WRONG! Retry (y/n)?");
      blit(buf, screen, 0, 0, 0, 0, 640, 480);
    }
    kp = readkey() >> 8;
  } while (kp == KEY_Y);
  return 0;
}
END_OF_MAIN();
```