Safe Cracker

Brush up on your "L337" speak. Safe Cracker is full of it. That's because when it was written haxor style was the theme. But never mind that, Safe Cracker is still a fun and challenging game.

You are given a limited amount of time to line up the tumblers within a small wedge. Use your mouse and click on the wheels with the right or left mouse button to speed them up or slow them down. Simply speeding them up won't always help. You need to speed them up until they are in line with other ones around them so you have time to make them all hit the wedge in the same moment.

As levels progress you're given more rings and a smaller wedge to challenge you. Challenge yourself and try to unlock the most safes you can.

If reading l337 is difficult for you, here's a handy guide:

- "54F3 |\|0" means "Safe No" (or safe number)
- "Ti|\/|3" means "Time"
- "53C0|\|65" / "Seconds"
- "U craX0red d4 54F3!!! ONE" / "You cracked the safe!!."
- "G4M3 0V3r LUUZAR!!!1ONE" / "Game over loser!"
- "PressORZ SpacORZ" / "Press Space."

Don't ask why, just enjoy the game.

Jakub Wasilewski wrote Safe Cracker for a Minor-Hack on May 26th, 2007.


```
/* safecracker.c listing begins: */

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

using namespace std;

/*****************************/

volatile int ticks = 0;
int safeNo = 1;
void tick()
{
  ticks++;
}

BITMAP *buffer;

/*****************************/

class Ring
{
  public:
    double angle;
    double minRot, maxRot;
    double rotSpeed;
    int   radius;

    BITMAP *spr;

    Ring(int radius, int width, double minRot, double maxRot)
      : minRot(minRot), maxRot(maxRot), radius(radius)
    {
      angle = (rand() / (double)RAND_MAX) * 2.0 * M_PI;
      rotSpeed = minRot + (rand() / (double)RAND_MAX) * (maxRot - minRot);

      spr = create_bitmap(2 * radius + 1, 2 * radius + 1);
      clear_to_color(spr, makecol(255, 0, 255));
      circlefill(spr, radius, radius, radius, makecol(100, 100, 100));
      circlefill(spr, radius, radius, radius-2, makecol(200, 200, 200));
      circlefill(spr, radius, radius, radius-width, makecol(100, 100, 100));
      circlefill(spr, radius, radius, radius-width-2, makecol(255, 0, 255));
      circlefill(spr, radius, width/2 - 1, width/2 - 3, makecol(0, 0, 255));
      circlefill(spr, radius + 2, width/2 + 1, width/2 - 3, makecol(0, 0, 0));
    }

    ~Ring()
    {
      destroy_bitmap(spr);
    }

    void draw(BITMAP *buffer)
    {
      rotate_sprite(buffer, spr, 400 - radius, 300 - radius,
        ftofix(angle * 128.0 / M_PI));
    }

    void update()
    {
      angle += rotSpeed * 0.01;
      while (angle > M_PI)
```
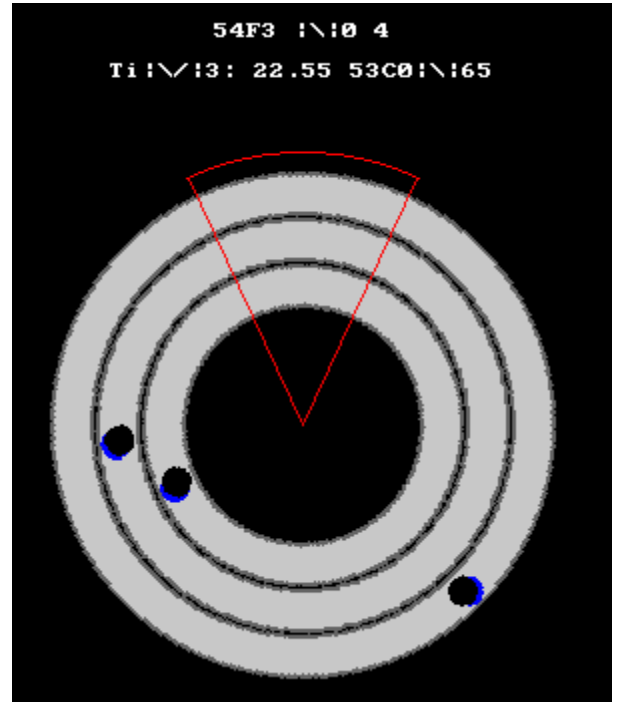
/ Listing continued on next page…*/*

```
/* Listing continued from previous page */
        angle -= 2 * M_PI;
      while (angle < -M_PI)
        angle += 2 * M_PI;

    }
};

void init()
{
  allegro_init();

  set_color_depth(32);
  set_gfx_mode(GFX_AUTODETECT_WINDOWED, 800, 600, 0, 0);

  install_keyboard();
  install_mouse();
  install_timer();

  install_int_ex(tick, BPS_TO_TIMER(100));

  buffer = create_bitmap(SCREEN_W, SCREEN_H);

  srand(time(NULL));
  show_mouse(screen);
}

int playLevel(Ring **rings, int ringCount, double maxTime, double tolerance)
{
  double time = maxTime;
  ticks = 0;
  bool end = false;
  bool completed = true;

  BITMAP *tolScr = create_bitmap(800, 600);
  clear_to_color(tolScr, makecol(255, 0, 255));

  double maxR = rings[ringCount-1]->radius + 10.0;

  line(tolScr, 400, 300, 400 + (int)(std::cos(M_PI / 2 - tolerance) * maxR),
    300 + (int)(-std::sin(M_PI/2 - tolerance) * maxR), makecol(255, 0, 0));
  line(tolScr, 400, 300, 400 + (int)(std::cos(M_PI / 2 + tolerance) * maxR),
    300 + (int)(-std::sin(M_PI/2 + tolerance) * maxR), makecol(255, 0, 0));
  arc(tolScr, 400, 300, ftofix((M_PI / 2 - tolerance) * 128.0 / M_PI),
    ftofix((M_PI/2 + tolerance) * 128.0 / M_PI), (int)maxR, makecol(255, 0, 0));

  while (!end)
  {
    while (ticks > 0)
    {
      int buttons = mouse_b;
      if (buttons)
      {
        int rng = -1;
        double dist = 100000.0, rdist = (mouse_x - 400) * (mouse_x - 400) +
          (mouse_y - 300) * (mouse_y - 300);
        rdist = std::sqrt(rdist);

        for(int i = 0; i < ringCount; i++)
          if (std::abs(rdist - rings[i]->radius + 10.0) < dist)
          {
            dist = std::abs(rdist - rings[i]->radius + 10.0);
            rng = i;
          }
```

/* Listing continued from previous page */

```cpp
        if (buttons & 1)
        {
          rings[rng]->rotSpeed += (rings[rng]->minRot-rings[rng]->maxRot) * 0.01;
          if (rings[rng]->rotSpeed < rings[rng]->minRot)
            rings[rng]->rotSpeed = rings[rng]->minRot;
        }
        if (buttons & 2)
        {
          rings[rng]->rotSpeed += (rings[rng]->maxRot-rings[rng]->minRot) * 0.01;
          if (rings[rng]->rotSpeed > rings[rng]->maxRot)
            rings[rng]->rotSpeed = rings[rng]->maxRot;
        }
        if (buttons & 4)
          allegro_message("%f %f", std::abs(rings[0]->angle), tolerance);
      }

      completed = true;
      for(int i = 0; i < ringCount; i++)
      {
        if (std::abs(rings[i]->angle) > tolerance)
          completed = false;
        rings[i]->update();
      }

      if (completed)
        return 1;

      time -= 0.01;

      if (time <= 0.0)
        return 0;

      ticks--;

      end = key[KEY_ESC];
      if (end)
        return -1;
    }

    clear_bitmap(buffer);
    for(int i = 0; i < ringCount; i++)
      rings[i]->draw(buffer);

    draw_sprite(buffer, tolScr, 0, 0);
    textprintf_centre_ex(buffer, font, 400, 100, makecol(255, 255, 255), -1,
      "54F3 |\\|0 %d", safeNo);
    textprintf_centre_ex(buffer, font, 400, 120, makecol(255, 255, 255), -1,
      "Ti|\\/|3: %0.2f 53C0|\\|65", time);

    blit(buffer, screen, 0, 0, 0, 0, SCREEN_W, SCREEN_H);
  }
}

int main()
{
  init();

  safeNo = 1;
  Ring *rings[5];
  int result;
  do
  {
```

/* Listing continued from previous page */

```
    int rc = (int)((safeNo + 1) / 4) + 2;
    if (rc > 4)
      rc = 4;
    for (int i = 0; i < rc; i++)
      rings[i] = new Ring(80 + 23 * i, 20, safeNo * 0.04, 2.0);
    result = playLevel(rings, rc, 25.0, 0.5 - safeNo * 0.015);
    if (result == -1)
      break;
    if (result == 1)
      textprintf_centre_ex(screen, font, 400, 40, makecol(255, 255, 0), -1,
        "U craX0red d4 54F3!!!ONE");
    else
      textprintf_centre_ex(screen, font, 400, 40, makecol(255, 255, 0), -1,
        "G4M3 0V3r LUUZAR!!!1ONE");
    textprintf_centre_ex(screen, font, 400, 50, makecol(255, 255, 0), -1,
      "PressORZ SpacORZ");
    int r;
    do
    {
      r = readkey() >> 8;
    } while (r != KEY_SPACE);

    safeNo++;
  } while (result == 1);
}
END_OF_MAIN();
```