Simon Fruit

Remember the sequence, repeat it back. The game should be somewhat familiar. Lights and sound help to cue you in to which one you need to remember.

This is another game that gives a simple example of how to accomplish sound using the allegro library. For that alone it would be worth a look. But once you give it a look, you may be hooked on trying to beat your high score.

Simon Fruit is by Jakub Wasilewski, submitted as an entry in the MinorHack challenge.

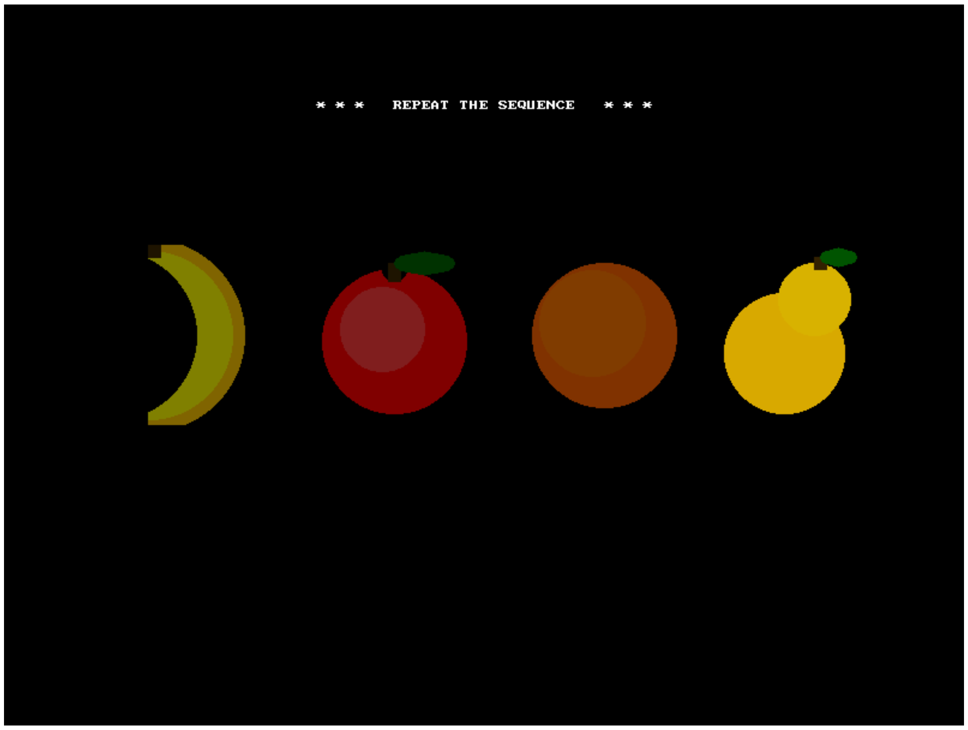
/* simonfruit.c listing begins: */

```
#include <allegro.h>
#include <vector>
#include <cmath>
#include <ctime>
#include <cstdlib>
using namespace std;

BITMAP *buf, *fruit[4], *arrow;

/**********************************/
volatile int ticks = 0;
static int sound_pitch[8], sound_time[8];

void mesg(int what, int ch, int first, int second = 0)
{
  unsigned char msg[3];
  msg[0] = what + ch;
  msg[1] = first;
  msg[2] = second;

  midi_out(msg, what < 0xc0 ? 3 : 2);
}

void tick()
{
  ticks++;
  for (int i = 0; i < 8; i++)
  {
    if (sound_time[i] > 0)
      sound_time[i]--;

    if (!sound_time[i])
      mesg(0x80, i, sound_pitch[i], 0);
  }
}

void sound(int prog, int pitch, int vol, int duration)
{
  static int channel = 0;
  channel = (channel + 1) % 8;

  mesg(0xC0, channel, prog);
  mesg(0x90, channel, pitch, vol);

  sound_pitch[channel] = pitch;
  sound_time[channel] = duration;
}

/**********************************/
void init()
{
  allegro_init();
  set_color_depth(32);
  set_gfx_mode(GFX_AUTODETECT_WINDOWED, 800, 600, 0, 0);

  install_timer();
  install_keyboard();
  install_sound(DIGI_AUTODETECT, MIDI_AUTODETECT, NULL);

  buf = create_bitmap(800, 600);
  for (int i = 0; i < 4; i++)
```

/* Listing continued on next page…*/

/* Listing continued from previous page */

```
  {
    fruit[i] = create_bitmap(150, 150);
    clear_to_color(fruit[i], makecol(255, 0, 255));
  }
  circlefill(fruit[0], 0, 75, 80, makecol(255, 200, 0));
  circlefill(fruit[0], 0, 75, 70, makecol(255, 255, 0));
  circlefill(fruit[0], -30, 75, 70, makecol(255, 0, 255));
  rectfill(fruit[0], 0, 0, 10, 10, makecol(60, 40, 0));

  circlefill(fruit[1], 75, 80, 60, makecol(255, 0, 0));
  circlefill(fruit[1], 65, 70, 35, makecol(255, 60, 60));
  circlefill(fruit[1], 75, 20, 10, makecol(255, 0, 255));
  rectfill(fruit[1], 70, 30, 80, 15, makecol(60, 40, 0));
  ellipsefill(fruit[1], 100, 15, 25, 10, makecol(0, 100, 0));

  circlefill(fruit[2], 75, 75, 60, makecol(255, 100, 0));
  circlefill(fruit[2], 65, 65, 44, makecol(255, 120, 0));

  circlefill(fruit[3], 50, 90, 50, makecol(255, 200, 0));
  circlefill(fruit[3], 75, 45, 30, makecol(255, 210, 0));
  rectfill(fruit[3], 75, 20, 85, 10, makecol(60, 40, 0));
  ellipsefill(fruit[3], 95, 10, 15, 8, makecol(0, 100, 0));

  arrow = create_bitmap(80, 80);
  clear_to_color(arrow, makecol(255, 0, 255));
  rectfill(arrow, 20, 20, 60, 80, makecol(200, 200, 200));
  rectfill(arrow, 15, 20, 55, 80, makecol(255, 255, 255));
  for (int i = 0; i < 5; i++)
    triangle(arrow, 40 - i, 0, 75 - i, 35, 5 - i, 35, makecol(200, 200, 200));
  triangle(arrow, 35, 0, 70, 35, 0, 35, makecol(255, 255, 255));

  install_int_ex(tick, BPS_TO_TIMER(100));

  srand(time(NULL));
}

/**********************************/
class Fruit
{
  public:

  int no;
  int litTime, blinked;
  int a;

  Fruit(int no = 0) : no(no)
  {
    litTime = 0;
    blinked = false;
  }

  void update()
  {
    if (litTime)
      litTime--;
    a -= 4;
  }

  void draw()
  {
    int l = (litTime) ? a : 128;
    if (l > 255) l = 255;
```



/* Listing continued on next page…*/

/* Listing continued from previous page */

```
    set_trans_blender(l, l, l, l);
    draw_trans_sprite(buf, fruit[no], 75 + (no ? 0 : 45) + 175 * no, 200);
  }
};

/*********************************/
int main()
{
  init();
  int ky;

do
{
  bool end = false;
  std::vector<int> sequence;

  Fruit f[4];
  int showingSequence = 0, doingSequence = 0;
  int arrowPos = 0;
  int waitTime = 0;

  for (int i = 0; i <4; i++)
    f[i] = Fruit(i);
  for (int i = 0; i < 3; i++)
    sequence.push_back(rand() % 4);

  ticks = 0;
  while (!end)
  {
    while (ticks > 0)
    {
      if (key[KEY_ESC])
        return 0;

      if ((showingSequence < sequence.size()))
      {
        int i = sequence[showingSequence];
        if (!f[i].blinked)
        {
          sound(55, 60 + i * 6, 80, 30);

          f[i].blinked = true;
          int t = 50 - sequence.size() * 2;
          if (t < 20) t = 20;
          f[i].litTime = t;
          f[i].a = 280;
        }
        else if (f[i].litTime == 0)
        {
          f[i].blinked = false;
          showingSequence++;
        }
      }
      else
      {
        while (keypressed())
        {
          int k = readkey() >> 8;
          if (k == KEY_LEFT)
            arrowPos = (arrowPos + 3) % 4;
          if (k == KEY_RIGHT)
            arrowPos = (arrowPos + 1) % 4;
          if ((k == KEY_SPACE) || (k == KEY_ENTER))
```

/* Listing continued on next page…*/

/* Listing continued from previous page */

```
          {
            if (arrowPos == sequence[doingSequence])
            {
              sound(55, 60 + arrowPos * 6, 80, 30);
              f[arrowPos].litTime = 20;
              f[arrowPos].a = 280;
              doingSequence++;
            }
            else
            {
              end = true;
            }
          }
        }

        if ((doingSequence == sequence.size()) && (waitTime == 0))
          waitTime = 50;
        if (waitTime > 0)
        {
          waitTime--;
          if (!waitTime)
          {
            sequence.push_back(rand() % 4);
            showingSequence = 0;
            doingSequence = 0;
            arrowPos = rand() % 4;
          }
        }
      }

      for (int i = 0; i < 4; i++)
        f[i].update();

      ticks--;
    }

    clear_bitmap(buf);
    for (int i = 0; i < 4; i++)
      f[i].draw();
    if (showingSequence == sequence.size())
      draw_sprite(buf, arrow, 125 + 175 * arrowPos, 450);

    textprintf_centre_ex(buf, font, 400, 80, makecol(255, 255, 255), -1,
      "* * *   REPEAT THE SEQUENCE   * * *");
    blit(buf, screen, 0, 0, 0, 0, 800, 600);
  }

  textprintf_centre_ex(screen, font, 400, 95, makecol(255, 255, 0), -1,
    "You managed to remember %d fruit.", sequence.size() - 1);
  textprintf_centre_ex(screen, font, 400, 105, makecol(255, 128, 0), -1,
    "[R] to restart, [ESC] to exit.");

  do{
    ky = readkey() >> 8;
    if (ky == KEY_ESC)
      return 0;
  } while ((ky != KEY_R) && (ky != KEY_ESC));
} while (ky != KEY_ESC);

return 0;
}
END_OF_MAIN();
```