Anaconda

Slide around eating fruit and avoiding your ever growing tail. Think you've done this one before? Think again. Instead of a snake which makes 90° turns this anaconda turns in smooth curves. Now how easy it is now?

If the game goes too fast for you adjust the number sent to `BPS_TO_TIMER()`. As the game progresses you'll find your snake moving faster, as well as turning wider. To stop this behavior comment out the line that reads `segLength += 0.001;`

Anaconda was written by Jakub Wasilewski for the MinorHack challenge on August 11th, 2007.

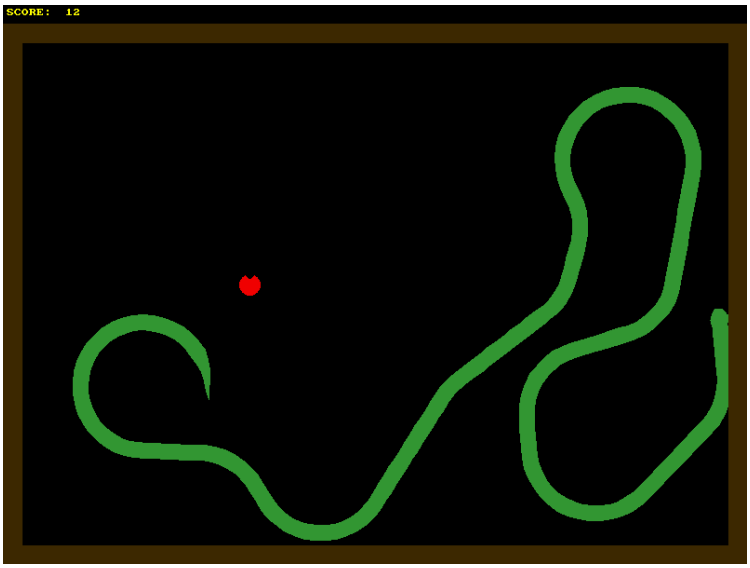
// Anaconda.ppc linsting begins:

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

BITMAP *apple, *buffer, *head;

volatile int ticks;
int snakeX, snakeY;

void tick()
{
  ticks++;
}
/************************************/
void init()
{
  allegro_init();

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

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

  apple = create_bitmap(24, 24);
  clear_to_color(apple, makecol(255, 0, 255));
  circlefill(apple, 12, 12, 11, makecol(240, 0, 0));
  int vert[] = {6, 0, 18, 0, 12, 5};
  polygon(apple, 3, vert, makecol(255, 0, 255));

  head = create_bitmap(24, 20);
  clear_to_color(head, makecol(255, 0, 255));
  ellipsefill(head, 12, 10, 12, 10, makecol(50, 150, 50));

  buffer = create_bitmap(800, 600);

  srand(time(NULL));
  install_int_ex(tick, BPS_TO_TIMER(100));
}

class Apple;
class Snake;
/************************************/
Apple *a;
Snake *s;

struct Segment
{
  double x, y;
  double angle;

  Segment(double x, double y, double angle) : x(x), y(y), angle(angle) {};

  static int v[8];

  static int *calcVertices(Segment s1, Segment s2, double w1, double w2)
  {
```

// Listing continued on next page...

// Listing continued from previous page

```
    v[0] = s1.x + cos(s1.angle + M_PI / 2) * w1;
    v[1] = s1.y - sin(s1.angle + M_PI / 2) * w1;
    v[2] = s1.x + cos(s1.angle - M_PI / 2) * w1;
    v[3] = s1.y - sin(s1.angle - M_PI / 2) * w1;
    v[4] = s2.x + cos(s2.angle - M_PI / 2) * w2;
    v[5] = s2.y - sin(s2.angle - M_PI / 2) * w2;
    v[6] = s2.x + cos(s2.angle + M_PI / 2) * w2;
    v[7] = s2.y - sin(s2.angle + M_PI / 2) * w2;
    return v;
  }
};
int Segment::v[8];

struct Snake
{
  double x, y;
  deque<Segment> seg;
  int longer;
  int moveCnt;
  double segLength, bearing;
  int score;

  Snake()
  {
    longer = 0; moveCnt = 2;
    segLength = 6.0;
    bearing = M_PI / 2;
    score = 0;

    for(int i = 0; i < 20; i++)
      seg.push_back(Segment(400, 250 + segLength * i, bearing));
  }

  void draw()
  {
    for (int i = 0; i < seg.size() - 6; i++)
    {
      polygon(buffer, 4, Segment::calcVertices(seg[i], seg[i+1], 8.0, 8.0),
        makecol(50, 150, 50));
    }
    for (int i = 0; i < 5; i++)
    {
      polygon(buffer, 4, Segment::calcVertices(seg[seg.size() - 6 + i],
      seg[seg.size() - 5 + i], 8.0 - i * 1.6, 6.4 - i * 1.6),
      makecol(50, 150, 50));
    }
  }

  void drawHead()
  {
    rotate_sprite(buffer, head, seg[0].x - 12, seg[0].y - 10,
    ftofix(256.0 - bearing * 128.0 / M_PI));
  }

  void update()
  {
    segLength += 0.001;

    if (key[KEY_LEFT]) bearing += 0.05;
    if (key[KEY_RIGHT]) bearing -= 0.05;

    moveCnt--;
    if (!moveCnt)
```



// Listing continued on next page...

// Listing continued from previous page

```
    {
      if (!longer)
        seg.pop_back();
      else
        longer--;

      double x = seg[0].x, y = seg[0].y;
      double nX = x + cos(bearing) * segLength, nY = y - sin(bearing)
        * segLength;
      seg.push_front(Segment(nX, nY, bearing));

      moveCnt = 3;
    }

    x = seg[0].x + cos(seg[0].angle) * 12.0;
    y = seg[0].y - sin(seg[0].angle) * 12.0;
  }
};

struct Apple
{
  public:
  double x, y;

  Apple(int x, int y) : x(x), y(y)
  {
  }

  void draw()
  {
    draw_sprite(buffer, apple, x - 12, y - 12);
  }

  bool update()
  {
    int diffX = s->x - x, diffY = s->y - y;
    return (diffX*diffX + diffY * diffY < 225);
  }
};
/***********************************/
int main()
{
  init();

  int k;
  do
  {
    bool end = false;
    a = new Apple(300, 300);
    s = new Snake();
    ticks = 0;

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

        if (a->update())
        {
```

// Listing continued on next page...

// Listing continued from previous page

```
          s->longer = 15;
          s->score++;
          delete a;
          int x = rand() % 640 + 80;
          int y = rand() % 440 + 90;
          a = new Apple(x, y);
        }
        s->update();
        ticks--;
        rest(0);
      }

      clear_bitmap(buffer);
      a->draw();
      s->draw();

      rectfill(buffer, 0, 20, 800, 40, makecol(60, 40, 0));
      rectfill(buffer, 0, 580, 800, 600, makecol(60, 40, 0));
      rectfill(buffer, 0, 20, 20, 600, makecol(60, 40, 0));
      rectfill(buffer, 780, 20, 800, 600, makecol(60, 40, 0));

      int col = getpixel(buffer, s->x, s->y);
      if (col == makecol(50, 150, 50) ||
        col == makecol(60, 40, 0))
          end = true;
      col = getpixel(buffer, s->x-2, s->y-2);
      if (col == makecol(50, 150, 50) ||
        col == makecol(60, 40, 0))
          end = true;
      col = getpixel(buffer, s->x+2, s->y-2);
      if (col == makecol(50, 150, 50) ||
        col == makecol(60, 40, 0))
          end = true;
      col = getpixel(buffer, s->x+2, s->y+2);
      if (col == makecol(50, 150, 50) ||
        col == makecol(60, 40, 0))
          end = true;
      col = getpixel(buffer, s->x-2, s->y+2);
      if (col == makecol(50, 150, 50) ||
        col == makecol(60, 40, 0))
          end = true;

      s->drawHead();
      textprintf_ex(buffer, font, 4, 4, makecol(255, 255, 0), -1,
        "SCORE: %3d", s->score);
      blit(buffer, screen, 0, 0, 0, 0, 800, 600);
    }

    rectfill(screen, 200, 200, 600, 300, makecol(0, 0, 0));
    textprintf_centre_ex(screen, font, 400, 240, makecol(255, 0, 0), -1,
      "* * *  G A M E   O V E R   * * *");
    textprintf_centre_ex(screen, font, 400, 260, makecol(255, 255, 0), -1,
      "FINAL SCORE: %3d", s->score);
    textprintf_centre_ex(screen, font, 400, 280, makecol(255, 127, 0), -1,
      "SPACE to restart, ESC to quit");

    do
    {
      k = readkey() >> 8;
    } while (k != KEY_ESC && k != KEY_SPACE);
  } while (k != KEY_ESC);

  return 0;
}
END_OF_MAIN();
```