Viscous Circles

The object of the game is to destroy as many ast...er...viscous circles as you can before their numbers over take you. You control a ship that can rotate left and right and move forward and back using the arrow keys and shoot using the space bar. When ever you destroy an astero...er...viscous circle it splits into two smaller parts which will split again and again until finally they are too small and can be destroyed. But don't be too complacent as their are more coming every few seconds.

The sides wrap so if you go off the left you come back on the right, but your shots don't, so it's generally a good idea to keep yourself in the middle of the screen dodging the growing swarm of as...viscous circles as long as you can.

Viscous circles was written by Jonatan Hedborg as an entry for the MinorHack challenge August 11th, 2007.

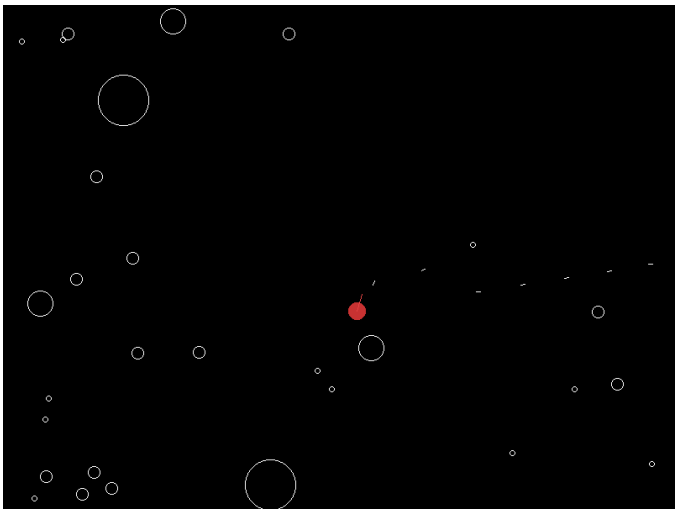
// viscouscircles.c linsting begins:

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

using namespace std;

volatile int counter;
void my_timer_handler()
{
  counter++;
}
END_OF_FUNCTION(my_timer_handler)

struct OBJECT {
  float x,y,vx,vy,angle,size;
  OBJECT(float _x, float _y, float _angle, float speed, float _size) {
    x = _x;
    y = _y;
    vx = cos(_angle)*speed;
    vy = sin(_angle)*speed;
    angle = _angle;
    size = _size;
  }
};

bool isHit(OBJECT *o1, OBJECT *o2) {
  if( (o1->x - o2->x)*(o1->x - o2->x) + (o1->y - o2->y)*(o1->y - o2->y)
    < (o1->size+o2->size)*(o1->size+o2->size) ) return true;
  else return false;
}

int main()
{
  srand(time(0));

  list<OBJECT> asteroids;
  list<OBJECT> shots;
  OBJECT player(400,300,0,0,10);

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

  install_keyboard();

  LOCK_VARIABLE(counter);
  LOCK_FUNCTION(my_timer_handler);

  install_int_ex(my_timer_handler,BPS_TO_TIMER(60));

  BITMAP *buffer = create_bitmap(800,600);
  int shot_delay = 0;
  int asteroid_delay = 100;
  bool gameOver = false;
  int killCount = 0;

  while(!key[KEY_ESC] && gameOver == false) {
    while(counter > 0) {
      counter--;
```

// Listing continued on next page...

// Listing continued from previous page

```
asteroid_delay--;
if( asteroid_delay < 0 ) {
  asteroid_delay = rand()%50+100;
  asteroids.push_back(
    OBJECT(3, rand()%500+50, (rand()%1000)/500.0-1, 3,30));
}

if( key[KEY_UP] || key[KEY_W] ) {
  player.vx += cos(player.angle)*0.7;
  player.vy += sin(player.angle)*0.7;
}
if( key[KEY_DOWN] || key[KEY_S]  ) {
  player.vx -= cos(player.angle)*0.7;
  player.vy -= sin(player.angle)*0.7;
}
if( key[KEY_LEFT] || key[KEY_A]  ) {
  player.angle -= 0.08;
}
if( key[KEY_RIGHT] || key[KEY_D]  ) {
  player.angle += 0.08;
}
if( key[KEY_SPACE] || key[KEY_M]  ) {
  shot_delay--;
  if( shot_delay < 0 ) {
    shots.push_back(OBJECT(player.x,player.y,player.angle,9,4));
    shot_delay = 5;
  }
}

player.vx -= player.vx * 0.1;
player.vy -= player.vy * 0.1;

player.x += player.vx;
player.y += player.vy;
if( player.x > 800 ) player.x = 0;
if( player.y > 600 ) player.y = 0;
if( player.x < 0 ) player.x = 800;
if( player.y < 0 ) player.y = 600;


for(list<OBJECT>::iterator it = shots.begin(); it!=shots.end();) {
  if( (*it).x > 800 || (*it).y > 600 || (*it).x < 0 || (*it).y < 0 ) {
    it = shots.erase(it);
  } else {
    bool kill = false;
    list<OBJECT>::iterator it2;
    for(it2 = asteroids.begin(); it2!=asteroids.end(); it2++) {
      if( isHit(&(*it),&(*it2)) ) {
        kill = true;
        break;
      }
    }

    if( kill ) {
      it = shots.erase(it);
      if( (*it2).size > 5 ) {
        asteroids.push_back(OBJECT((*it2).x,(*it2).y,
          (rand()%1000)/1000.0*M_PI*2,3,(*it2).size/2));
        asteroids.push_back(OBJECT((*it2).x,(*it2).y,
          (rand()%1000)/1000.0*M_PI*2,3,(*it2).size/2));
      }
      asteroids.erase(it2);
```



// Listing continued on next page...

// Listing continued from previous page

```
            killCount++;
          } else {
            (*it).x += (*it).vx;
            (*it).y += (*it).vy;
            it++;
          }
        }
      }

      For (list<OBJECT>::iterator it = asteroids.begin();
        it!=asteroids.end(); it++) {

        if( (*it).x > 800 ) (*it).x = 0;
        if( (*it).y > 600 ) (*it).y = 0;
        if( (*it).x < 0 ) (*it).x = 800;
        if( (*it).y < 0 ) (*it).y = 600;

        if( isHit(&player,&(*it)) ) gameOver = true;

        (*it).x += (*it).vx;
        (*it).y += (*it).vy;
      }
    }

    clear(buffer);
    circlefill(buffer,player.x,player.y,player.size,makecol(200,50,50));
    line(buffer,player.x,player.y,player.x+cos(player.angle)*20,
      player.y+sin(player.angle)*20,makecol(230,70,70));

    for(list<OBJECT>::iterator it = shots.begin(); it!=shots.end(); it++) {
      line(buffer,(*it).x,(*it).y,(*it).x+cos((*it).angle)*5,
        (*it).y+sin((*it).angle)*5,makecol(255,255,255));
    }

    for(list<OBJECT>::iterator it = asteroids.begin();
      it!=asteroids.end(); it++) {

      circle(buffer,(*it).x,(*it).y,(*it).size,makecol(255,255,255));
    }

    vsync();
    blit(buffer,screen,0,0,0,0, 800,600);
  }

  clear(screen);
  textprintf_centre_ex(screen, font, 400, 300, makecol(180,180,180),-1,
    "GAME OVER! You destroyed %i aster...vicious circles! (press enter)",
    killCount);
  while(!key[KEY_ENTER]) { rest(1); }

      return 0;
}
END_OF_MAIN()
```