**SECTION 2. Applied mathematics. Mathematical modeling.**

**Shevtsov Alexandr Nikolayevich**
candidate of technical Sciences,
President of International Academy of Theoretical & Applied Sciences, Kazakhstan

# ABOUT SOME ALGORITHMS FOR CONSTRUCTING THREE-DIMENSIONAL DYNAMICAL MODELS

***Abstract***: *This article presents The algorithms on Delphi, to build a mathematical model of the motion of particles, their interaction with each other and the environment.*
***Key words***: *algorithms, model, Delphi.*

# О НЕКОТОРЫХ АЛГОРИТМАХ ПОСТРОЕНИЯ ТРЕХМЕРНЫХ ДИНАМИЧЕСКИХ МОДЕЛЕЙ

***Аннотация***: *В данной статье приведены алгоритмы на Дельфи, для построения математической модели движения частиц, их взаимодействия друг с другом и внешней средой.*
***Ключевые слова***: *алгоритмы, модель, дельфи.*

Существует довольно много задач гидро и газо-динамики, механики, нанотехнологии и других областей - требующих построения трехмерной модели и исследования поведения отдельных ее элементов или частиц, во взаимодействии и динамическом изменении во времени. К примеру Премия Гордона Белла 2013 года, вручаемая ежегодно за достижения в области высокопроизводительных вычислений, досталась группе исследователей, которые смоделировали возникновение и эволюцию пузырьков жидкости при кавитации (Рис.1). Группа специалистов из Швейцарского федерального технологического института, исследовательского центра IBM, Технического университета Мюнхена и Ливерморской национальной лаборатории в США получила приз в 10 тысяч долларов за свои работы о моделированию кавитации. Ученым удалось получить наиболее подробную модель, которая рассматривает поведение 15 тысяч пузырьков в жидкости [1].

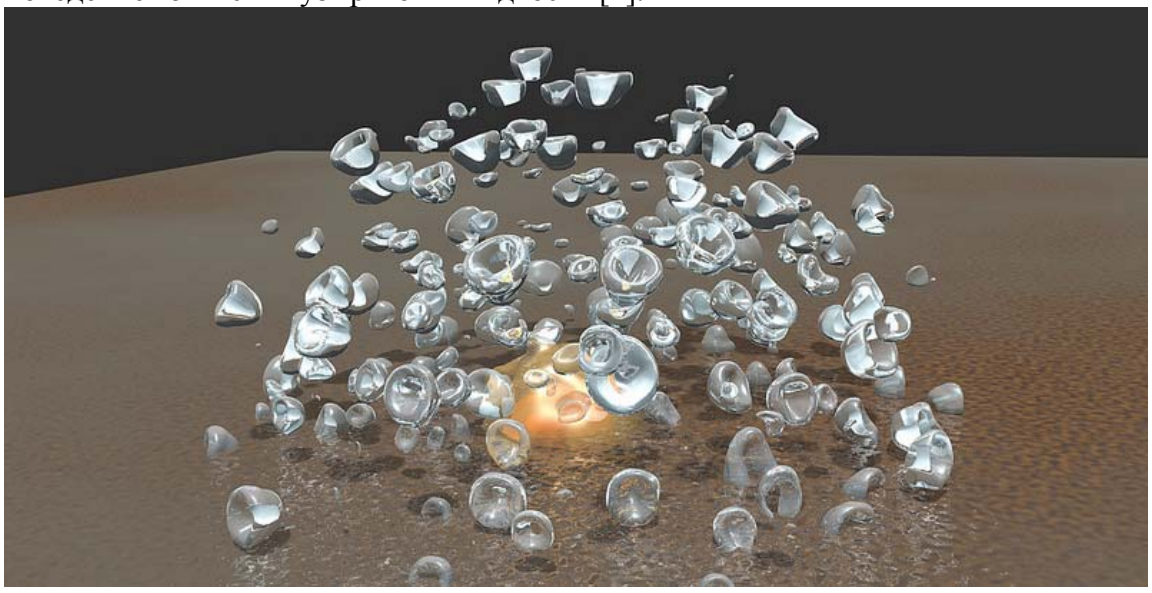

**Рисунок 1 - Визуализация исследования кавитационных пузырьков[2].**

Рассмотрим основы построения подобной модели и алгоритмы на Delphi для ее практической реализации. Введем некоторые упрощения:

- будем рассматривать частицы (атомы) как центральное ядро диаметром 0,1анг. и внешний диаметр (самого атома, на котором начинается взаимодействие с окружающими частицами) порядка 1 анг.
- зададим ограничения внешнего объема - 10 анг.$^3$
- количество от 1 до 1000 частиц.
- зададим массив с динамическими данными, для каждого из них, пространственную координату, единичный вектор скорости и уровень кинетической энергии в пределах [0…1].
- введем ограничение подвижности, при 30% энергии, и менее, частица будет прекращать свое движение.
- зададим законы столкновения частиц, характерные для жидких или газообразных сред.
- а также граничные условия, и начальные уровни энергий.

Создадим приложение и форму (Рис.2.). Построение и отрисовку всех элементов будем производить через DirectX и бибилиотеку OpenGL.

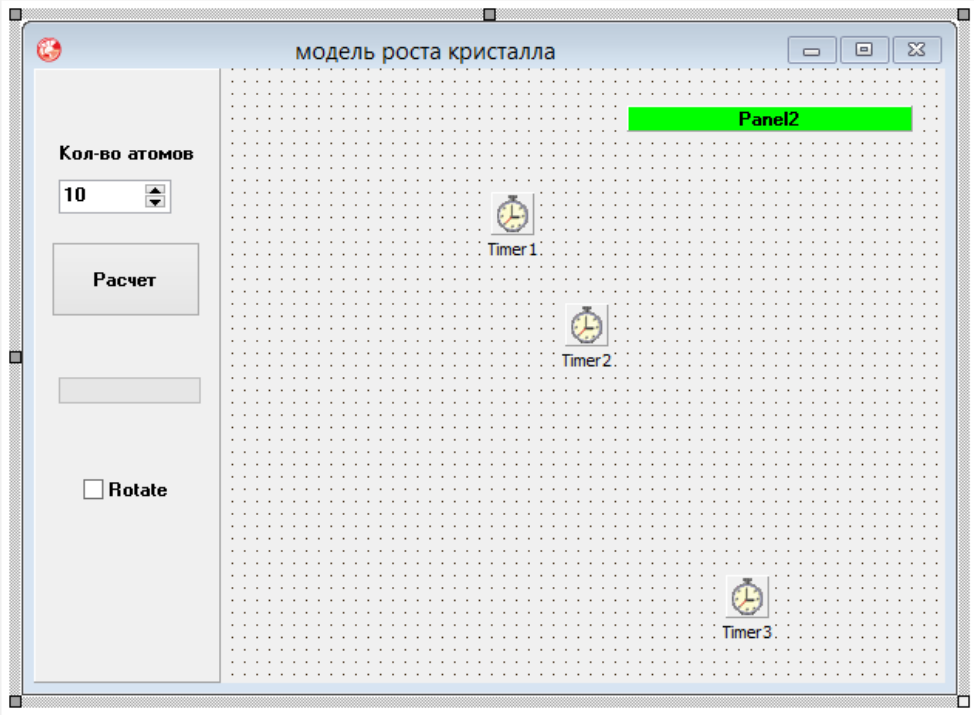


**Рисунок 2 – Окно программы.**

**code Delphi**

```
unit Unit1;
interface
uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, ExtCtrls, OpenGL, ComCtrls, StdCtrls, Spin;
type
  TForm1 = class(TForm)
    Timer1: TTimer;
    Timer2: TTimer;
    Panel1: TPanel;
```

```
    SpinEdit1: TSpinEdit;
    Button1: TButton;
    Label1: TLabel;
    Panel2: TPanel;
    ProgressBar1: TProgressBar;
    Timer3: TTimer;
    CheckBox1: TCheckBox;
    procedure FormCreate(Sender: TObject);
    procedure FormResize(Sender: TObject);
    procedure Timer1Timer(Sender: TObject);
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
    procedure Timer2Timer(Sender: TObject);
    procedure FormDblClick(Sender: TObject);
    procedure SpinEdit1Change(Sender: TObject);
    procedure Button1Click(Sender: TObject);
    procedure FormDestroy(Sender: TObject);
  private
    ghRC:HGLRC;
    ghDC:HDC;
    procedure Draw;
  public
    { Public declarations }
  end;
type aa=array[1..1000,-3..3] of double;
var
  Form1: TForm1;
dd:double;
n,i,j,k,box:integer;
a:aa;
var
  quadObj: GLUquadricObj;
implementation

{$R *.dfm}
function bSetupPixelFormat(DC:HDC):boolean;
var
    pfd:PIXELFORMATDESCRIPTOR;
    ppfd:PPIXELFORMATDESCRIPTOR;
    pixelformat:integer;
begin
    ppfd := @pfd;
    ppfd.nSize := sizeof(PIXELFORMATDESCRIPTOR);
    ppfd.nVersion := 1;
    ppfd.dwFlags :=  PFD_DRAW_TO_WINDOW xor
                PFD_SUPPORT_OPENGL xor
                PFD_DOUBLEBUFFER;
    ppfd.dwLayerMask := PFD_MAIN_PLANE;
    ppfd.iPixelType := PFD_TYPE_RGBA;
    ppfd.cColorBits := 16;
    ppfd.cDepthBits := 16;
```

```
    ppfd.cAccumBits := 0;
    ppfd.cStencilBits := 0;
    pixelformat := ChoosePixelFormat(dc, ppfd);
    if pixelformat=0 then
    begin
       MessageBox(0, 'ChoosePixelFormat failed', 'Error', MB_OK);
       bSetupPixelFormat:=FALSE;
       exit;
    end;

    if SetPixelFormat(dc, pixelformat, ppfd)=false then
    begin
       MessageBox(0, 'SetPixelFormat failed', 'Error', MB_OK);
       bSetupPixelFormat:=FALSE;
       exit;
    end;
    bSetupPixelFormat:=TRUE;
end;

procedure TForm1.FormCreate(Sender: TObject);
var
  p: TGLArrayf4;
  d: TGLArrayf3;
 I,j: Integer;
begin
box:=10;
   n:=1;
        dd:=1;
  ghDC := GetDC(Handle);
  if bSetupPixelFormat(ghDC)=false then
    Close();
  ghRC := wglCreateContext(ghDC);
  wglMakeCurrent(ghDC, ghRC);

  glClearColor(4.0, 4.0, 4.0, 4.0);
  FormResize(Sender);
  glEnable(GL_COLOR_MATERIAL);
  glEnable(GL_DEPTH_TEST);
  glEnable(GL_LIGHTING);
  glEnable(GL_LIGHT0);
    p[0]:=3;
    p[1]:=3;
    p[2]:=3;
    p[3]:=0;
    d[0]:=0;
    d[1]:=0;
    d[2]:=-3;
  glLightfv(GL_LIGHT0,GL_POSITION,@p);
  glLightfv(GL_LIGHT0,GL_SPOT_DIRECTION,@d);
```

```
  for I := 1 to 1000 do
 begin
 for j := 1 to 3 do    a[i,j]:=random(100)/10;
 for j := -3 to -1 do    a[i,j]:=(random(200)-100)/100;
a[i,0]:=1;
 end;
quadObj := gluNewQuadric;
    end;

procedure TForm1.FormDestroy(Sender: TObject);
begin
 gluDeleteQuadric(quadObj);
end;

procedure TForm1.FormResize(Sender: TObject);
var kr:integer;
begin
  glViewport( 0, 0, Width, Height );
  glMatrixMode( GL_PROJECTION );
  glLoadIdentity();
  kr:=15;
  glOrtho(-kr,kr,    -kr,kr,    -kr,kr);
  gluLookAt( 5,5,5,    0,0,0,    0,0,10);
  glMatrixMode( GL_MODELVIEW );

panel2.Width:=trunc(10*((width-panel1.Width)/(2*kr))/1);
panel2.Caption:='10  ангстрем';

n:=SpinEdit1.Value;
end;

procedure TForm1.SpinEdit1Change(Sender: TObject);
begin  n:=SpinEdit1.Value;  end;

procedure TForm1.Button1Click(Sender: TObject);
begin
n:=SpinEdit1.Value;
progressbar1.Max:=trunc(n/10000);
 quadObj := gluNewQuadric;
  Draw;
  Timer1.Enabled:=true;
end;

procedure TForm1.Draw;
var s:double;
begin
for I := 1 to n do
begin
for j := 1 to 3 do
begin
```

```
if a[i,0]>0.3 then
// движение
a[i,j]:=a[i,j]+a[i,-j]*a[i,0] else
begin
//алгоритм кристаллизации
end;

if a[i,j]>box then begin a[i,-j]:=-abs(a[i,-j]); a[i,j]:=box; end;
if a[i,j]<0 then begin a[i,-j]:=abs(a[i,-j]);   a[i,j]:=0 end;
end;
if a[i,1]=0 then  a[i,0]:=0.9*abs(a[i,0]);

for j := 1 to n do  if i<>j then
begin
s:=sqrt(sqr(a[i,1]-a[j,1])+sqr(a[i,2]-a[j,2])+sqr(a[i,3]-a[j,3]));
if s<2 then
begin
s:=(a[i,0]+a[j,0])/2;
a[i,0]:=s;
a[j,0]:=s;

for k := -3 to -1 do
begin
s:=a[i,k];
a[i,k]:=a[j,k];
a[j,k]:=s;
end;
end; end; end;
glPopMatrix();

  glClear(GL_DEPTH_BUFFER_BIT xor GL_COLOR_BUFFER_BIT);
  glColor3f(0,0,1);
glPushMatrix();
for I := 1 to n do
begin
if a[i,0]>0.3 then glColor3f(a[i,0],0,0) else
glColor3f(a[i,0],0,1-a[i,0]);
glPushMatrix();
glTranslatef(a[i,1],a[i,2],a[i,3]);     //смещение центра
gluSphere(quadObj, 0.1,         10,10);
glPopMatrix();
end;

 glPopMatrix();
glBegin(GL_LINE_STRIP);
 glcolor3f(50, 0,0);
 glVertex3f(50, 0,0);
 glVertex3f(0, 0,0);

 glcolor3f(0, 10,0);
```

```
 glVertex3f(0, 50,0);
 glVertex3f(0, 0,0);

 glcolor3f(0, 0,10);
 glVertex(0, 0, 50);
glEnd;
if checkbox1.checked then glRotatef(-1,       0,0,1);
  SwapBuffers(ghDC);
end;

procedure TForm1.Timer1Timer(Sender: TObject);
begin
draw;
end;

procedure TForm1.Timer2Timer(Sender: TObject);
begin
timer2.Enabled:=false;
form1.WindowState:=wsmaximized;
end;

procedure TForm1.FormClose(Sender: TObject; var Action: TCloseAction);
begin
  if ghRC<>0 then
  begin
    wglMakeCurrent(ghDC,0);
    wglDeleteContext(ghRC);
  end;
  if ghDC<>0 then
    ReleaseDC(Handle, ghDC);
end;
end.
```

В результате получим следующую визуализацию, без наложения текстур на триангулированные поверхности частиц (Рис.3). Модель динамическая и управляется несколькими таймерами, определяющими временной шаг расчета взаимодействий и координат. Модель делает расчет в реальном времени, и при увеличении числа частиц, более 100, компьютер (оценка производительности 5,7) начинает притормаживать. Как следствие более целесообразным будет разбиение задачи на: выполнение всех расчетов без прорисовки, и отдельным алгоритмом прорисовка элементов в реальном времени.

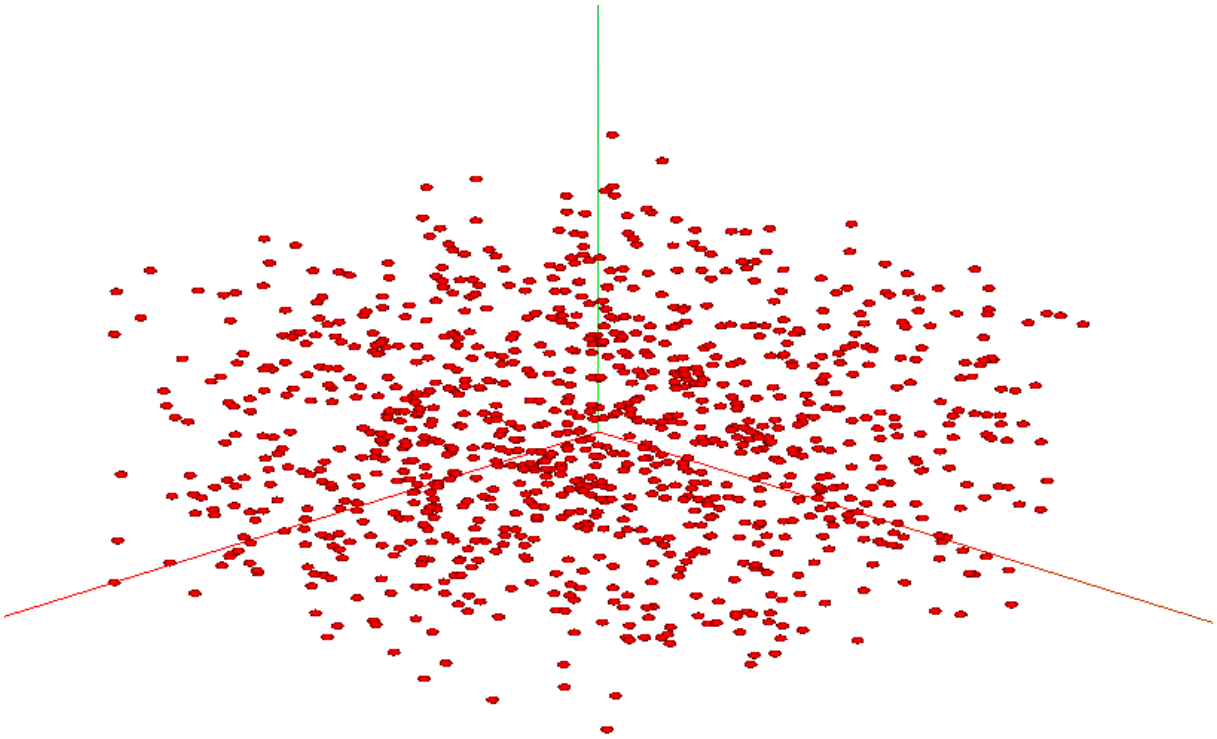

**Рисунок 3 – Модель в движении.**

## Литература

1. Премию Гордона Белла дали за моделирование пузырьков. [Электронный ресурс]. URL: http://lenta.ru/news/2013/11/25/bellsprize/ (дата обращения: 29.12.2013).
2. Моделирование кавитационных пузырьков получило премию Гордона Белла. [Электронный ресурс]. URL: http://habrahabr.ru/company/ibm/blog/205800/ (дата обращения: 29.12.2013).