DESENVOLVIMENTO DE UM SOFTWARE DE GERAÇÃO E VISUALIZAÇÃO DE NANOESTRUTURAS

Aluno: Marcos Paulo Moraes
Orientador: André Silva Pimentel

Introdução

A nanotecnologia está associada a diversas áreas de pesquisa e produção na escala nano (escala atômica) tais como a medicina, eletrônica, ciência da computação, física, química, biologia e engenharia dos materiais. Seu princípio básico é a construção de estruturas e novos materiais a partir dos átomos que são os tijolos básicos da natureza. Algumas das ferramentas mais importantes em Nanotecnologia são os programas computacionais para a geração, visualização e simulação de novos nanomateriais a nível molecular. Hoje em dia, existem vários programas computacionais deste tipo que funcionam apropriadamente em moléculas razoavelmente pequenas. No entanto, são escassos os programas específicos para a aplicação em moléculas grandes encontradas na Nanotecnologia. No Brasil, não existe uma única empresa de softwares ou um único software para gerar e visualizar nanomateriais. Portanto, este projeto visa a desenvolver um software único no mercado brasileiro e fomentar a recursos humanos nesta área interdisciplinar tão carente no Brasil.

Objetivos

O objetivo deste projeto é desenvolver um programa computacional para geração e visualização de estruturas químicas importantes na Nanotecnologia. O programa, escrito em C e Visual Basic (VB.NET), possibilitará ao usuário gerar e visualizar estruturas de grafenos, nanotubos, cristais, zeólitas e fulerenos. As manipulações de computação gráfica serão desenvolvidas utilizando a linguagem C e a API OpenGL. Tem-se como meta desenvolver e registrar este software em um prazo de 2 a 3 anos.

Metodologia

No projeto, tem-se desenvolvido um programa, escrito com a linguagem C, para construir nanoestruturas tais como grafenos, nanotubos, cristais, fulerenos e zeólitas. Ao passo que a interatividade do humano com o computador é implementada utilizando a linguagem Visual Basic.

As nanoestruturas de carbono, como grafenos e nanotubos, são geradas utilizando algoritmos de replicação, translação e rotação. Os grafenos são estruturas planares, em geral, formadas por átomos de carbono localizados nos vértices de hexágonos. Os hexágonos são criados através de rotação e replicação no eixo xy.

Os nanotubos de carbono são estruturas cilindricas, sendo geradas através da rotação dos átomos de carbono dos grafenos, previamente construídos, através de um vetor quiral, representado pelos índices (n, m). Estes índices denotam o número de vétores unitários ao longo de duas direções no retículo cristalino do grafeno [1]. Se $m = 0$, os nanotubos são chamados de “zigzag”. Ao passo que, se $n = m$, eles são chamados de “braço de cadeira”. Ao contrário, os nanotubos são chamados de “quirais”.

Os fulerenos são usualmente gerados utilizando o algoritmo espiral da Teoria de Grafos encontrado na literatura. [2] Neste projeto, os fulerenos estão sendo simplesmente construídos a partir de um banco de dados de fulerenos regulares e irregulares de até 720 carbonos produzido por este algoritmo.

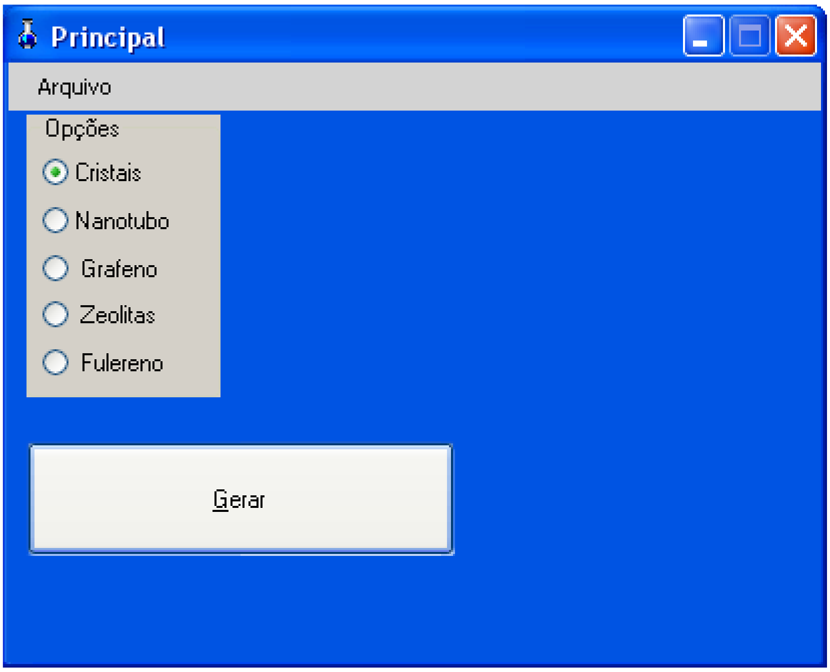
As nanoestruturas de cristais e zeólitas são construídas a partir de bancos de dados cristalográficos existentes na literatura. [3,4] Nestas bases de dados tem-se a célula unitária, a menor parte de um cristal que contém suas características. Esta unidade é repetida tridimensionalmente através de operações de translação utilizando os vetores primitivos e as coordenadas das redes de Bravais, que caracterizam a estrutura atômica de um cristal. [3]

Similarmente, a base de dados cristalográfica das zeólitas contem as coordenadas de seus átomos no modelo T [4], que pode ser utilizada para construir a célula unitária através de operações de translação e rotação. Para este projeto, uma base de dados foi gerada com a célula unitária de cada zeólita e esta foi replicada utilizando os vetores de Bravais.

Utilizando técnicas de computação gráfica, foi implementado, também, um programa capaz de gerar a visualização dos arquivos de saída dos programas citados acima. Utilizando a linguagem C e a API (Interface de Programação de Aplicativos) OpenGL, esta aplicação é capaz de rotacionar, ampliar (zoom in) e reduzir (zoom out) as nanoestruturas utilizando o mouse do computador. Também é possível obter as distâncias entre os átomos, os ângulos entre átomos escolhidos e ângulos diedros.

A aplicação principal, que integra todos os subprogramas e provê a interação com o usuário, foi desenvolvida utilizando a linguagem Visual Basic. Através dela pode-se utilizar cada gerador e visualizador.

Podemos ver na figura abaixo o menu principal do programa:



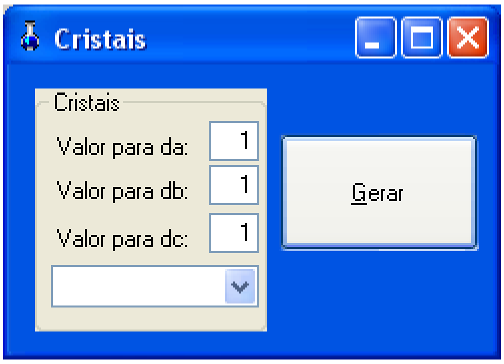
A janela principal tem os seguintes menus: Arquivo, onde podemos acessar a janela de visualização de estruturas; e Fechar, que encerra o aplicativo. Temos também as opções de geração: Cristais, Nanotubo, Grafeno, Zeólitas e Fulereno.

No projeto, é utilizada a estrutura (struct) *atomo*, criando um novo tipo chamado *Atomo*. Esse novo tipo é capaz de conter as coordenadas de um átomo no espaço e o símbolo do elemento. Abaixo, a declaração da struct:

```
struct atomo
{
     char elem[3];
     long float x;
     long float y;
     long float z;
};
typedef struct atomo Atomo;
```

A 'janela geradora de cristais', vista na figura abaixo, contém os seguintes campos:

- Valor para da: valor inteiro referente ao quanto a molécula será replicada na direção X.
- Valor para db: valor inteiro referente ao quanto a molécula será replicada na direção Y.
- Valor para dc: valor inteiro referente ao quanto a molécula será replicada na direção Z.
- Uma lista com todos os arquivos de células unitárias existentes no sistema.
- Um botão *Gerar* que, ao ser clicado, executa a replicação do cristal escolhido nas direções pretendidas.

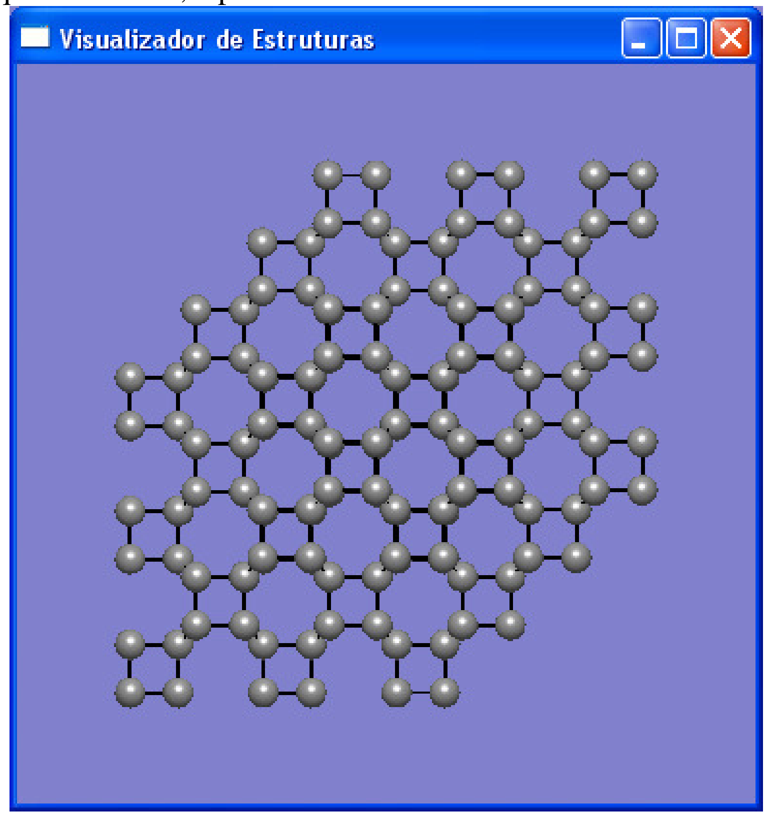


O programa gerador de cristais recebe o nome do arquivo contendo a célula unitária e os valores para da, db e dc. Tendo os valores e o acesso ao arquivo, o programa salva as coordenadas presentes em um vetor do tipo *Atomo*. Salva também as coordenadas dos vetores primitivos, que define as direções (eixos primitivos) da rede de Bravais. Pode-se, então, replicar a célula unitária nas três direções desejadas, através do código abaixo.

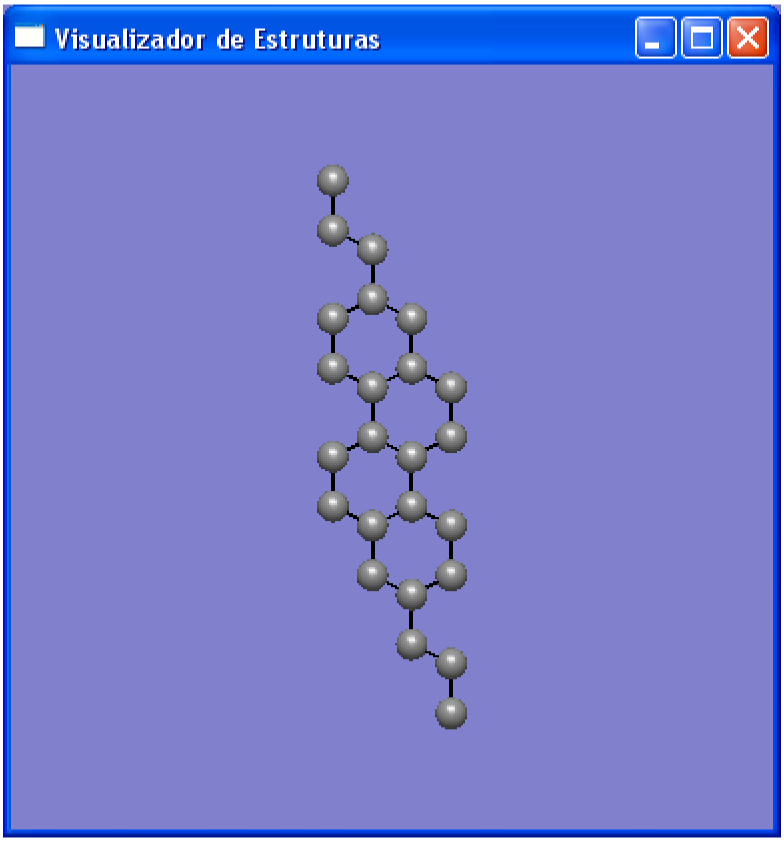
```
for(i=0;i<da;i++)
{
     for(j=0;j<db;j++)
     {
          for(k=0;k<dc;k++)
          {
               for(m=0;m<l;m++)
               {
                    strcpy(res[p].elem,coords[m].elem);
                    res[p].x = coords[m].x_CC + ((i)*a[0].x + (j)*a[1].x + (k)*a[2].x);
                    res[p].y = coords[m].y_CC +  ((i)*a[0].y + (j)*a[1].y + (k)*a[2].y);
                    res[p].z = coords[m].z_CC + ((i)*a[0].z + (j)*a[1].z + (k)*a[2].z);
                    p++;
               }

          }
     }
}
```

Onde o vetor *a* é o vetor primitivo e o vetor *coords* são as coordenadas da célula unitária. O vetor *resp* será o vetor com o cristal replicado. Após a replicação, o cristal resultante é salvo em um arquivo de saída, formatado com as especificações de um arquivo *.gjf* (Gaussian Input Format).

Com o arquivo de saída, é possível visualizar o cristal resultante.

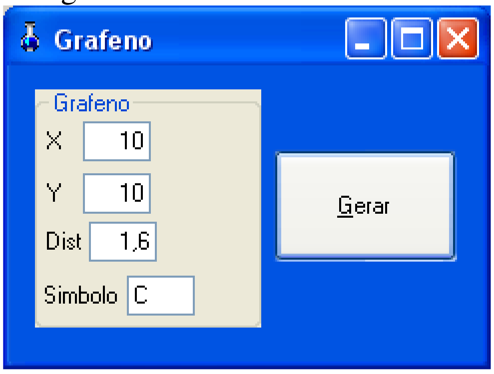


Exemplo de saída da replicação de um cristal.



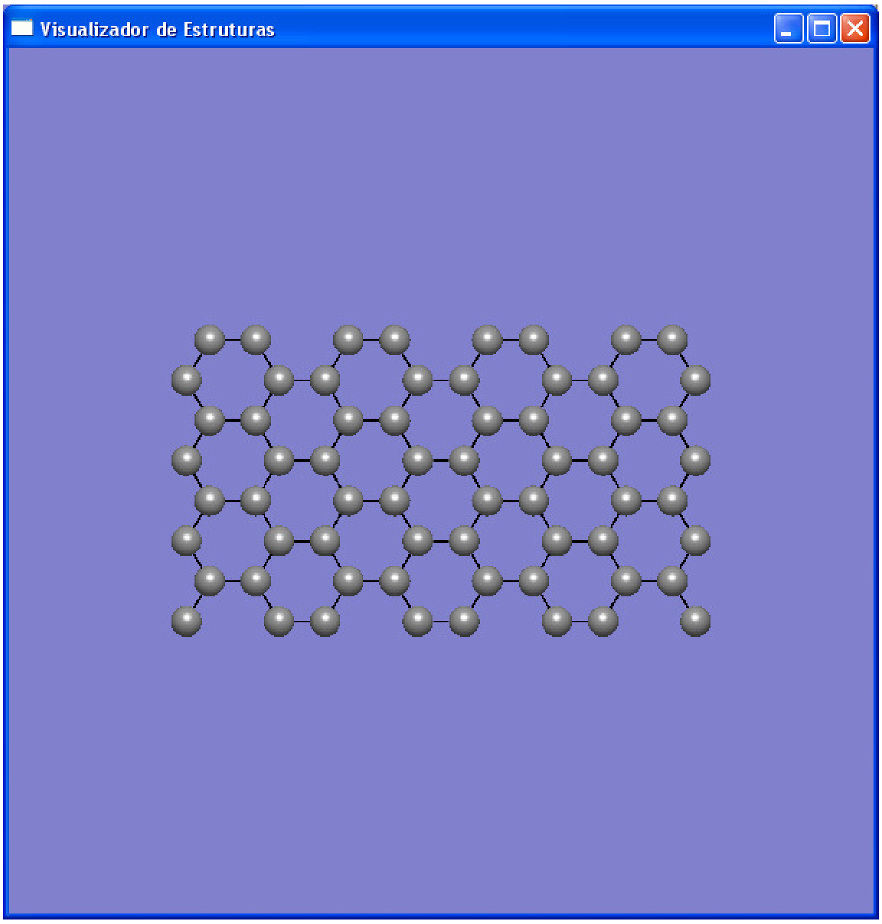
O mesmo cristal visto de lado.

A geração de grafenos é dada da seguinte forma: o usuário escolhe o elemento do qual o grafeno será composto, a distância entre os elementos, o número de átomos no eixo X e o número de átomos do eixo Y do grafeno.

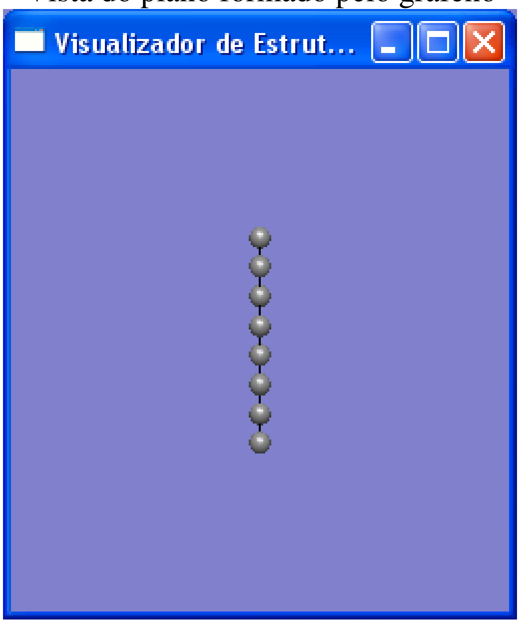


Com os dados, o gerador do grafeno cria um hexágono cujos vértices são átomos do elemento escolhido e o replica na direção X e Y até atingir o limite definido pelo usuário.

```
/*Definindo um hexágono*/
hex[0].x=dist;
hex[0].y=0.0;
hex[0].z=0.0;
for (i = 1; i<6; i++)
{
    hex[i].x = hex[i-1].x*cos(pi/3) - hex[i-1].y*sin(pi/3);
    hex[i].y = hex[i-1].x*sin(pi/3) + hex[i-1].y*cos(pi/3);
    hex[i].z = 0.0;
}
```



Vista do plano formado pelo grafeno



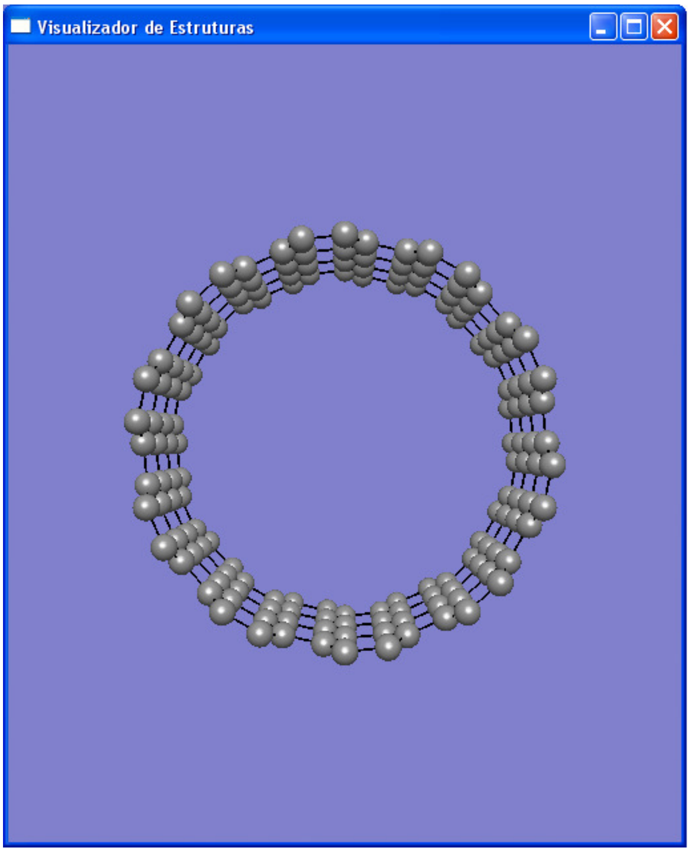
Vista lateral do grafeno

A ‘janela geradora de nanotubos’, vista na figura abaixo, contém os seguintes campos:

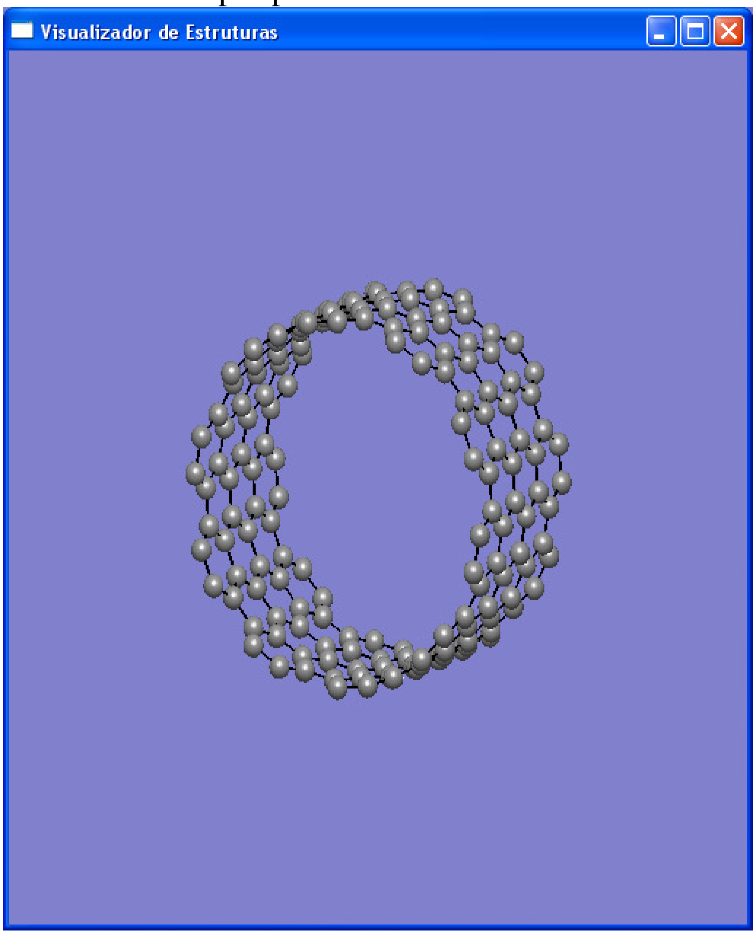
- Nº de paredes existentes no nanotubo;
- O valor quiral N para cada parede do nanotubo;
- O valor quiral M para cada parede do nanotubo;
- Deformação radial do nanotubo: 0 se não for requerida nenhuma deformação;
- Número de translações: valor referente ao comprimento do nanotubo;

Como visto anteriormente, é o vetor quiral que denomina o tipo do nanotubo. Caso n = m, teremos o tipo Armchair. Se m = 0, o nanotubo será zigzag. Se n ≠ m, teremos o tipo quiral.

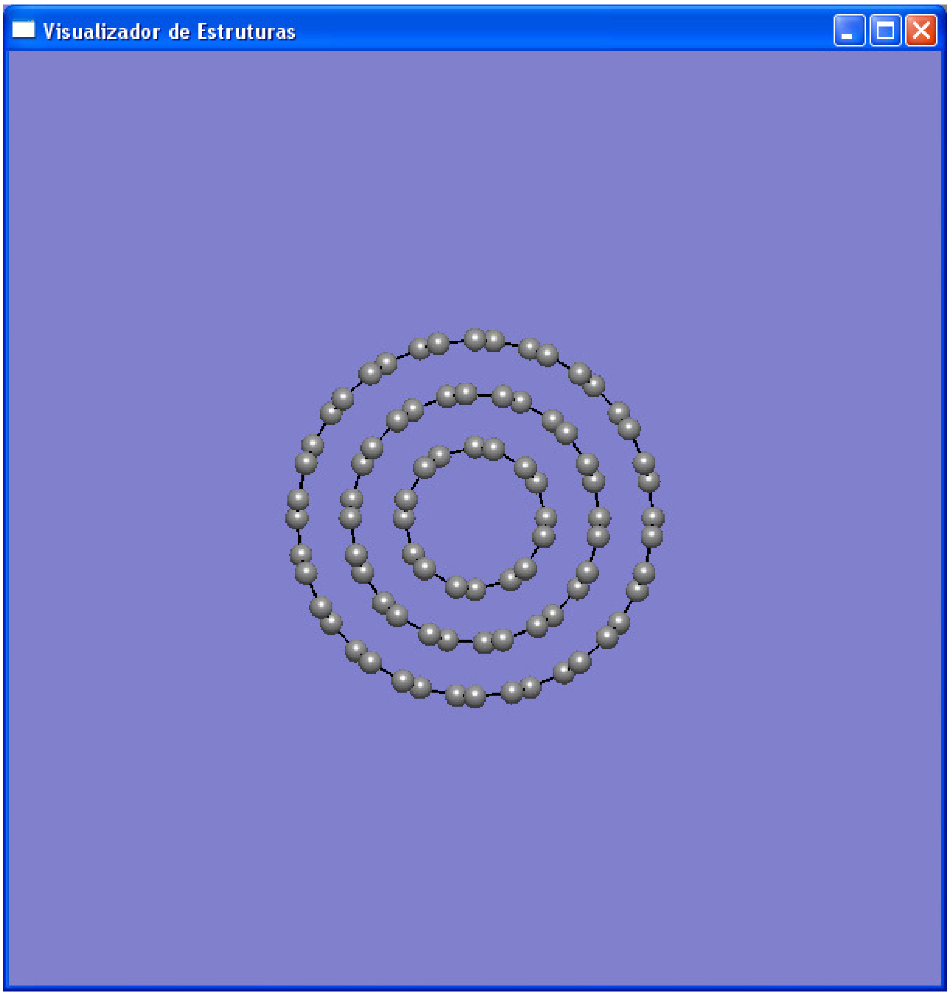
A geração de nanotubo é similar ao de grafeno. Mas, após fazer o grafeno, a estrutura plana sofre uma deformação (de acordo com o vetor quiral) a fim de se tornar cilíndrica, dando origem ao nanotubo.



Nanotubo em visão perspectiva.

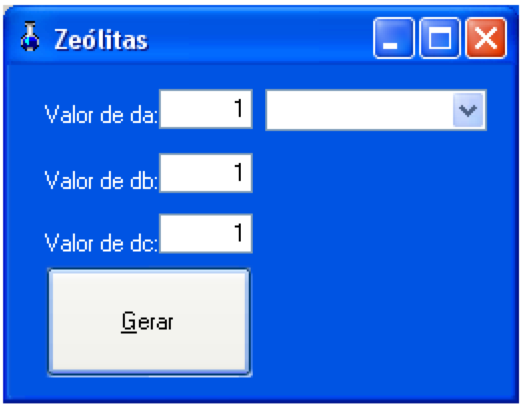


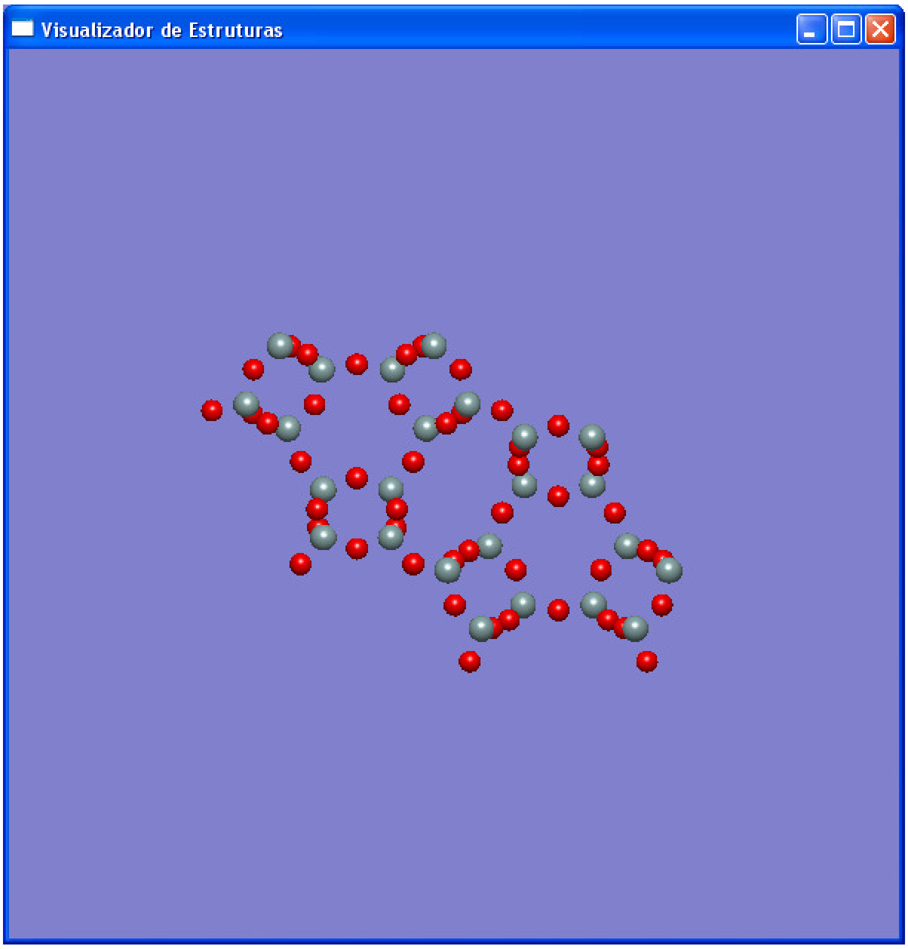
Nanotubo visto inclinado, em visão ortogonal.



Nanotubo de múltiplas paredes (MWNT), visão ortogonal.

As zeólitas, como escrito anteriormente, foram geradas através de uma base de dados com a sua célula unitária e replicada como o cristal.

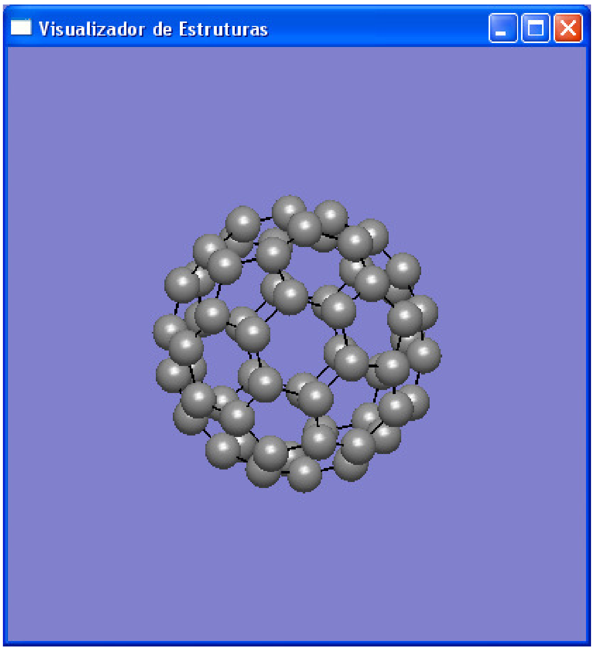




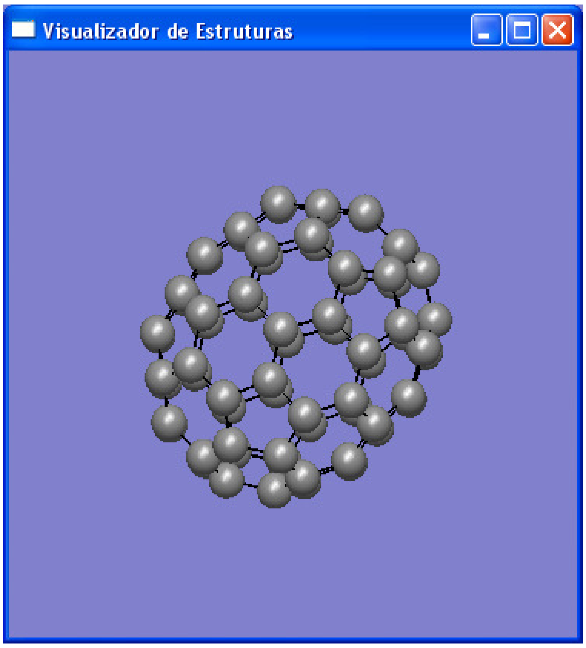
Visualização de uma zeólita

A geração de fulerenos é feita pela conversão de uma biblioteca com mais de 1000 fulerenos, regulares e irregulares. O usuário escolhe um dos arquivos e é gerado um arquivo *.gjf* com as coordenadas do fulereno. É possível modificar o elemento do fulereno.

Gerador de Fulerenos



Fulereno Regular com 60 Carbonos



Fulereno Irregular com 70 carbonos

O Programa de Visualização de Estruturas, feito em C e utilizando a API OpenGL, é um executável que recebe o nome do arquivo a ser visualizado e cria em uma janela o desenho da estrutura. O programa foi gerado em rotinas de fácil entendimento, cada uma referente a uma função específica do projeto.

- Read_File: função encarregada da leitura das coordenadas dos átomos presentes no arquivo. Salva as coordenadas em um vetor da estrutura Atomo;
- Tipo_Atomo: função que verifica se determinado elemento é um átomo e define uma cor referente a ele e seu raio atômico, encontrado na literatura;
- Calcula_Ângulo: função que calcula o ângulo formado entre 3 átomos;
- Calcula_Ângulo_Diedro: função que calcula o ângulo diedro formado entre 4 átomos;
- Verifica_Distância: função que calcula a distância entre dois átomos;
- Ligação: verifica se um átomo, pela distância, tem ligação com outro. As ligações, apenas para efeito de visualização, serão sempre simples.

```
void tipo_atomo(char* Elemento,float* raio)
{
    if(strcmp(Elemento,"O")==0)
    {
        *raio=0.50;
        glColor3f(0.898f, 0.0f, 0.0f);
    }
    else if(strcmp(Elemento,"H")==0)
    {
        *raio=0.3;
        glColor3f(0.8f, 0.8f, 0.8f);
    }
    else if(strcmp(Elemento,"C")==0)
    {
        *raio=0.70;
        glColor3f(0.557f, 0.557f, 0.557f);
    }
```

Trecho do código da função que verifica o elemento

O OpenGL, por sua vez, é programado em rotinas de *callback*, funções que são chamadas sempre que um evento (clique no mouse, arrasto de objetos, digitação e outros) acontece.

- Display: função encarregada de desenhar a estrutura na tela, é chamada sempre que há movimentação do mouse ou quando outras funções, auxiliares, são executadas;
- Mouse e Motion: funções que controlam as ações feitas pelo mouse como, por exemplo, rotação (com o botão esquerdo do mouse), zoom in e zoom out (com o botão do meio). Essas funções são facilmente modificadas, podendo aumentar ou diminuir a velocidade com que o objeto é rotacionado, ampliado ou reduzido;
- Keyboard: função por onde o usuário, por meio do teclado, tem a capacidade de modificar átomos, calcular ângulos, diedros, escolher o modo de visualização;

Há, também, funções referentes a inicialização da janela de visualização, que controla a luz, a cor de fundo, a transparência dos elementos. Os modos de visualização são dois: ortogonal ou perspectiva. Cabe ao usuário decidir o modo que desejar.

Conclusões

Tendo o software produzido neste projeto a capacidade de criar e visualizar nanoestruturas com número elevado de átomos, pode-se afirmar que o mesmo supre as necessidades expostas.

Há também a possibilidade de implementar novas rotinas que criam outras estruturas, como polímeros, sem a necessidade de alteração no código dos demais.

O programa gerador de cristais pode ser desenvolvido de outra maneira, utilizando a simetria existente em sua base de dados.

O layout do programa é provisório, sendo modificado de acordo com as necessidades futuras do projeto.

O programa é de fácil manuseio, podendo ser utilizado em sala de aula no ensino médio e na universidade, como um método mais interativo de ensino e aprendizagem, ou com a finalidade de preparar moléculas para simulação molecular em pesquisas acadêmicas. Percebe-se que estas ferramentas didáticas podem ser mais efetivas na aprendizagem do aluno, atraindo mais a atenção do aluno em sala de aula.

Referências

1 - http://en.wikipedia.org/wiki/Carbon_nanotube. Acesso em: Setembro 2009

2 - P. W. Fowler e K. M. Rogers *J. Chem. Inf. Comput. Sci.* **2001,** *41,* 108

3 – Base de dados dos Cristais: http://cst-www.nrl.navy.mil/lattice/ Acesso em: Setembro 2009

4 – Base de dados das zeólitas: http://www.iza-structure.org/databases/ Acesso em: Outubro 2009