SUBMITTED BY: BREITENBERGER, NICODEMUS

1 Exercise: Rotation (Ball)

```
#include <BALL/FORMAT/PDBFile.h>
#include <BALL/KERNEL/system.h>
#include <BALL/KERNEL/chain.h>
#include <BALL/KERNEL/residue.h>
#include <BALL/KERNEL/atom.h>
#include <BALL/KERNEL/protein.h>
#include <BALL/MATHS/angle.h>

int main(int argc, char* argv[]){

   BALL::Vector3 xAxisVector(1,0,0);
   BALL::Angle angle = BALL::Angle(BALL::Constants::PI/3, true);
   BALL::System translationSystem;
   BALL::PDBFile sourceFile;
   BALL::PDBFile rotatedFile;
   BALL::Matrix4x4 rotationMatrix;
   rotationMatrix.setRotation(angle, xAxisVector);

   //Sanity checks for command-line arguments
   if(argc == 3){

      sourceFile.open(argv[1], std::ios::in);
      rotatedFile.open(argv[2], std::ios::out);

   }else{

      std::cout << "Wrong amount of Parameters\n\n Useage: prog
    inFile outFile\n";
      return 1;

   }

   if(sourceFile.is_open()){

      //the read action for a pdbfile reads the data into a system
     which can be manipulated afterwards - and saved
      sourceFile.selectModel(1);
      sourceFile.read(translationSystem);
      sourceFile.close();

      // iterate over all proteins
      for(BALL::MoleculeIterator m_it = translationSystem.
    beginMolecule(); +m_it; ++m_it){
```

```
41
42          //We need to check if the molecule we just grabbed with
    m_it is indeed a protein. if it is not,
43          //We cannot iterate over chains and residues.
44          if (BALL::RTTI::isKindOf<BALL::Protein>(*(m_it))){
45
46              BALL::Protein* protein = BALL::RTTI::castTo<BALL::
    Protein>(*(m_it));
47
48              if(protein->countChains() > 0){
49
50                for(BALL::ChainIterator ch_it = protein->beginChain
    (); +ch_it; ++ch_it){
51
52                   for(BALL::ResidueIterator r_it = ch_it->
    beginResidue(); +r_it; ++r_it){
53
54                      for(BALL::AtomIterator a_it = r_it->beginAtom
    (); +a_it; ++a_it){
55
56                         //Rotating Atom-Positions; it may be
    necessary to move them to origin, rotate and put them back
57                         //but since are single Atoms it should not
     matter
58                         a_it->setPosition(rotationMatrix * a_it->
    getPosition());
59
60                      }
61                   }
62                }
63             }
64          }
65       }
66
67       //writing to out-file
68       if(rotatedFile.is_open()){
69
70          rotatedFile.write(translationSystem);
71
72       }else{
73
74             std::cout << "Could not write to system" << '\n';
75             return 1;
76
77       }
78
79    }else{
80
81       std::cout << "Could not open PDB file." << '\n';
```

```
82        return 1;
83
84    }
85 }
```

2 Exercise: Creation of a molecule (Ball)

```
1  #include <BALL/KERNEL/atom.h>
2  #include <BALL/KERNEL/bond.h>
3  #include <BALL/KERNEL/PTE.h>
4  #include <BALL/KERNEL/molecule.h>
5  #include <BALL/MATHS/vector3.h>
6  #include <BALL/MATHS/angle.h>
7  #include <BALL/FORMAT/PDBFile.h>
8
9  int main(int argc, char* argv[]){
10
11   BALL::PDBFile outFile;
12
13   //Sanity checks for command-line arguments
14   if(argc == 2){
15
16       outFile.open(argv[1], std::ios::out);
17
18   }else{
19
20       std::cout << "Wrong amount of Parameters\n\n Useage: prog
       outFile\n";
21       return 1;
22
23    }
24
25     BALL::Angle phi, theta, phi2, phi3;
26     BALL::Atom* nitrogen = new BALL::Atom;//(BALL::PTE[Element::
       OXYGEN]);
27     BALL::Atom* hydro1 = new BALL::Atom;
28     BALL::Atom* hydro2 = new BALL::Atom;
29     BALL::Atom* hydro3 = new BALL::Atom;
30
31   phi = BALL::Angle(2*M_PI/3, true); //to ensure that the molecule
       is planar
32   theta = BALL::Angle(0, true); //angle to
33   phi2 = BALL::Angle(4*M_PI/3, true); //to ensure that the
       molecule is planar
```

```
34    phi3 = BALL::Angle(2*M_PI/3, true); //to ensure that the
        molecule is planar
35
36    BALL::Vector3 nitroPos(0, 0, 0); // nitrogen placed in origin
37    BALL::Vector3 hdyro1_1Pos(101.7/100, phi, theta); //101.7/100 =
        distance in angstrom
38    BALL::Vector3 hdyro2_1Pos(101.7/100, theta, phi2);
39    BALL::Vector3 hdyro3_1Pos(101.7/100, theta, phi3);
40
41    nitrogen->setElement(BALL::PTE[BALL::Element::NITROGEN]);
42    nitrogen->setPosition(nitroPos);
43    hydro1->setElement(BALL::PTE[BALL::Element::HYDROGEN]);
44    hydro1->setPosition(hdyro1_1Pos);
45    hydro2->setElement(BALL::PTE[BALL::Element::HYDROGEN]);
46    hydro2->setPosition(hdyro2_1Pos);
47    hydro3->setElement(BALL::PTE[BALL::Element::HYDROGEN]);
48    hydro3->setPosition(hdyro3_1Pos);
49
50    BALL::Molecule* h3n = new BALL::Molecule;
51    BALL::Bond* n_f = hydro1->createBond(*nitrogen);
52    BALL::Bond* n_s = hydro2->createBond(*nitrogen);
53    BALL::Bond* n_t = hydro3->createBond(*nitrogen);
54
55    h3n->append(*nitrogen);
56    h3n->append(*hydro1);
57    h3n->append(*hydro2);
58    h3n->append(*hydro3);
59
60    BALL::System systemHN3("H3N");
61
62    systemHN3.append(*h3n);
63
64    if(outFile.is_open()){
65
66      outFile.write(systemHN3);
67
68    }else{
69
70      std::cout << "Could not write to file" << argv[2] << '\n';
71
72    }
73
74    return 0;
75  }
```

3 Exercise: The ∇ operator

$$\vec{F} = [3x + y]\vec{i} + [z + 3x]\vec{j} + [x + y + z]\vec{k}$$

Divergence of vector $\vec{F}$:

$$\nabla \cdot \vec{F} = \partial_x F_x + \partial_y F_y + \partial_z F_z$$

$$= \partial_x[3x + y] + \partial_y[z + 3x] + \partial_z[x + y + z]$$

$$= 3 + 0 + 1 = 4$$

Rotation of vector $\vec{F}$:

$$\nabla \times \vec{F} = \begin{pmatrix} \partial_y F_z - \partial_z F_y \\ \partial_z F_x - \partial_x F_z \\ \partial_x F_y - \partial_y F_x \end{pmatrix} = \begin{pmatrix} 1 - 1 \\ 0 - 1 \\ 3 - 1 \end{pmatrix} = \begin{pmatrix} 0 \\ -1 \\ 2 \end{pmatrix}$$

$$V = \frac{1}{(x^2 + y^2 + z^2)^3} - \frac{1}{(x^2 + y^2 + z^2)^3}$$

$$\Rightarrow V = 0$$

Gradient of scalar V:

$$\nabla V = 0$$

Divergence of a scalar is not defined $\nabla \cdot V$ as well as the rotation $\nabla \times V$.

4 Exercise: Calculation of a potential

$$\vec{F} = -\vec{\nabla} V$$

$$\begin{pmatrix} F_x \\ F_y \\ F_z \end{pmatrix} = - \begin{pmatrix} \partial_x V \\ \partial_y V \\ \partial_z V \end{pmatrix}$$

$$\vec{F} = [6x - y\sin(xy)]\vec{i} + [z - x\sin(xy)]\vec{j} + \left[y + \frac{1}{1 + z^2}\right]\vec{k}$$

$$F_x = 6x - y\sin(xy)$$

$$F_y = z - x\sin(xy)$$

$$F_z = y + \frac{1}{1 + z^2}$$

$$V = -\int F_x \,\mathrm{d}x + C(y,z)$$

$$= -\int 6x - y\sin(xy)\,\mathrm{d}x + C(y,z)$$

$$= -(3x^2 + \cos(xy)) + C'(y,z)$$

$$= -3x^2 - \cos(xy) - \int F_y \,\mathrm{d}y + C''(z)$$

$$= -3x^2 - \cos(xy) - yz + C''(z)$$

$$V = -3x^2 - \cos(xy) - yz - \arctan(z) + c$$

with c a constant according to x,y, and z