1 Описание модуля

Модуль `geometry` представляет из себя подключаемый пакет на языке C++. Состоит из двух файлов `geometry.hpp` и `geometry.cpp`. В модуле реализован набор классов, отвечающих за базовые пространственные фигуры (примитивы), преобразование фигур (сдвиг, поворот, трансформация) и алгебры битовых операций для комбинирования нескольких фигур (объединение, пересечение, исключение). Модуль использует в качестве подключаемого пакета библиотеку aivlib. Проекты с подключенным модулем можно также компилировать с использованием SWIG 2.

В основе пакета лежит технология «Конструктивная сплошная геометрия» (Constructive Solid Geometry, CSG) [1]. Характерный пример использования этой технологии изображен на рис. 1.

Все классы объектов являются наследниками базового типа `BaseFigure`. Из наследников класса набирается древовидное дерево выражения, листьями которго являются примитивы, а остальные вершины либо объекты преобразований либо объекты битовых операций. Для контроля за указателями использован прокси-класс `Figure`, с "умным"указателем в `private` поле `std::shared_ptr<BaseFigure> figure`.

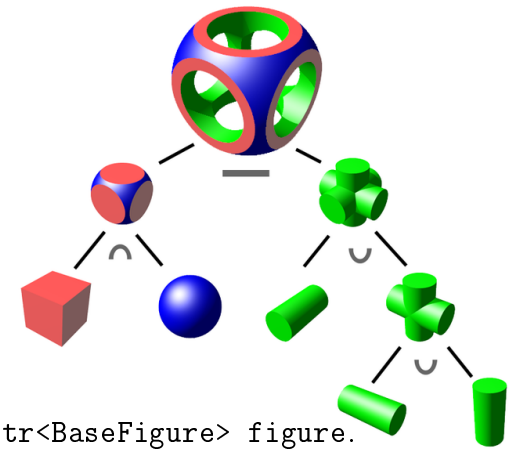

Рис. 1: Сложный объект может быть представлен двоичным деревом, где «листья» — это объекты, а узлы — операции. (∩ — пересечение, ∪ — объединение, — — разность)

У каждого класса фигуры определены два метода — `get_min()` и `get_max()`, которые возвращают координаты левого нижнего и правого верхнего угла вмещающего прямоугольного параллелепипида(bounding box). При преобразовании фигур и при битовых операциях bounding box вычисляется относительно детей.

2 Примитвы

Примитивы —основы для набора сложных фингур. Для создания экземпляров прокси-класса для примитивов используются порождающие глобальные функции.

2.1 Сфера

Создается порождающей глобальной функцией

```
Figure spheroid(const aiv::vctr<3> center, double R);
```

, где `center` — центр сферы, `R` — радиус сферы.

2.2 Цилиндр

Создается порождающей глобальной функцией

```
Figure cylinder(const aiv::vctr<3> bottom_origin_center, const aiv::vctr<3> &n,
    double R, double H);
```

, где `bottom_origin_center` — центр нижнего основания цилиндра, `n` — направляющаяя, выходящяя из нижнего основания в сторону верхнего, перпендикулярно нижнему основанию, `R` —радиус оснований, `H` — высота цилиндра.

2.3 Прямоугольный параллелепипид

Создается порождающей глобальной функцией

```
Figure box(const aiv::vctr<3> bottom_origin_center, const aiv::vctr<3> &n,
    double phi, double A, double B, double H);
```

, где `bottom_origin_center` — центр нижней грани, `n` — направляющаяя, выходящяя из нижней грани в сторону верхней, перпендикулярно нижней грани, `phi` — угол поворота вокруг линии, соединяющей центры нижней и верхней граней по часовой стрелке, `A`,`B`,`C` — длина ребер вдоль осей X,Y,Z соответственно.

2.4 Куб

И функцией

```
Figure cube(const aiv::vctr<3> bottom_origin_center, const aiv::vctr<3> &n,
    double phi, double A);
```

, где `bottom_origin_center` — центр нижней грани, `n` — направляющаяя, выходящяя из нижней грани в сторону верхней, перпендикулярно нижней грани, `phi` — угол поворота вокруг линии, соединяющей центры нижней и верхней граней по часовой стрелке, `A` — длина ребра куба.

3 Преобразование фигур

Преобразования фигур реализованы как методы класса `Figure`, возвращающие экземпляры класса `Figure`.

3.1 Сдвиг

```
Figure Figure::move(const aiv::vctr<3> &offset);
```

, где `offset` — смещение фигуры.

3.2 Поворот вокруг прямой по часовой стрелке

```
Figure Figure::rotate(const aiv::vctr<3> &center, const aiv::vctr<3> &n_phi);
```

, где `center` — точка, из которой будет выходит образующая прямой, вокруг которой поворачиваем, `n_phi` — направление образующей прямой, вокруг которой поворачиваем, а длина вектора `n_phi` равна углу, на который поворачиваем.

3.3 Трансформация

```
Figure Figure::transform(const aiv::vctr<3> &center, const aiv::vctr<3> &ox,
    const aiv::vctr<3> &oy, const aiv::vctr<3> &oz);
```

, где `center` — центр новой системы координат в координатах старой,`ox`, `oy`, `oz` — новые оси координат, выраженные в координатах старой системы.

4 Алгебра битовых операций

Битовые операции над фигурами исполнены при помощи перегрузки операторов сложения, умножения и вычитания для экземпляров класс `Figure`. Нужно быть внимательным и правильно расставлять скобки, так как битовых операциях не всегда выполняется коммутативность.

- Объединение — перегруженный оператор сложения $(+)$
- Пересечение — перегруженный оператор умножения $(*)$
- Исключение — перегруженный оператор вычитания $(-)$

5 Пример использования

5.1 C++

```
#include "geometry.hpp"
using namespace aiv;
int main(){
   Figure f1 = cylinder(Vctr(0.,0.,0.),Vctr(0.,0.,1.), 20., 20.);
   Figure f2 = f1.rotate(Vctr(0.,0.,0.), Vctr(1.,1.,1));
   Figure f3 = f1 + f2;

   vctr<3> my_min = f3.get_min()
   vctr<3> my_max = f3.get_max()

   printf("Результат проверки вхождения вектора Vctr(3., 10., 0.) в
    фигуру f3 %s", f3.check(Vctr(3., 10., 0.)) ? "true" : "false")
   return 0;
}
```

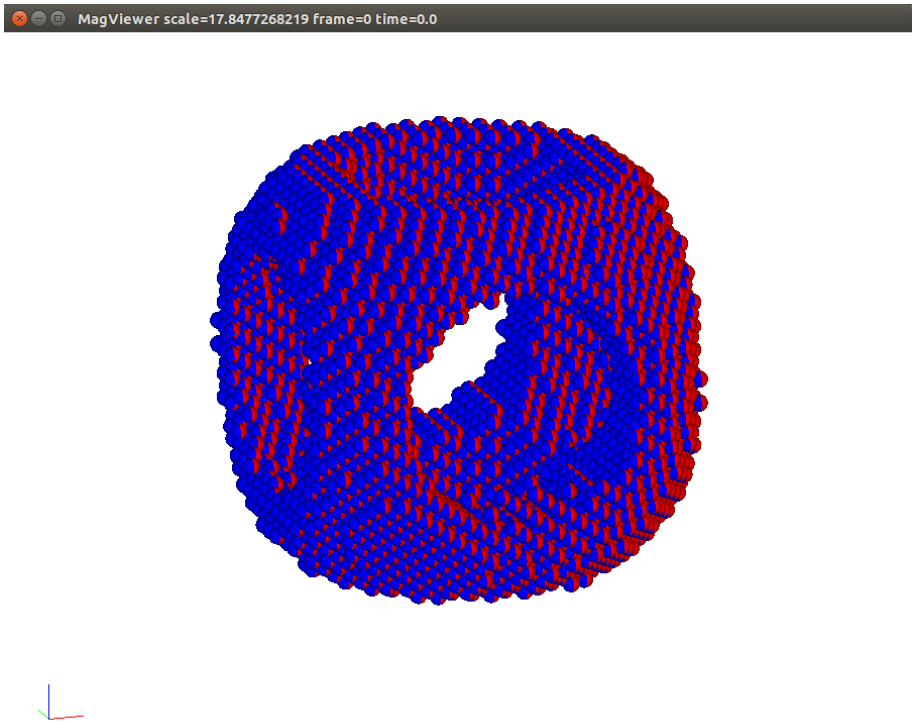

Рис. 2: Полученный магнитный образец с применением модуля

5.2 python

```
from geometry import *
import math
e_fig1 = cube(Vctr(0., 0., 0.), Vctr(0.,0.,1.), 0., 20)
e_fig2 = spheroid(Vctr(0.,0., 10.), 10 * math.sqrt(2)*0.9)
e_fig3 = cylinder(Vctr(0.,0.,0.), Vctr(0., 0., 1.), 6, 20 )
e_fig4 = cylinder(Vctr(-10.,0.,10.), Vctr(1., 0., 0.), 6, 20 )
e_fig5 = cylinder(Vctr(0.,-10.,10.), Vctr(0., 1., 0.), 6, 20 )

final_fig = e_fig1*e_fig2-(e_fig3+e_fig4+e_fig5)
final_fig_1 = e_fig1*e_fig2-(e_fig3+e_fig4+
    e_fig5.move(Vctr(3.,0.,3.)).rotate(Vctr(3.,-11.,3.), Vctr(0.,0.,-0.2)))
```

5.3 magnus

В пакете magnus для моделирования эволюции магнетков модуль `geometry` играет одну из главных ролей. При помощи фигур из этого модуля задается формы области магнитных образцов. В примере использования на языке Python фигура `final_fig` является прототипом фигуры с рисунка 1. Если проинициализировать магнитный образей при помощи этой фигуры и вывести его изображение через пакет MagView, то получим аналогиную картину (см. рис. 2).

Список литературы

[1] *Foley, James D.* (1996), "12.7 Constructive Solid Geometry Computer Graphics: Principles and Practice, Addison-Wesley Professional, pp. 557–558, ISBN 9780201848403.