Geometry

A CPP Template Class
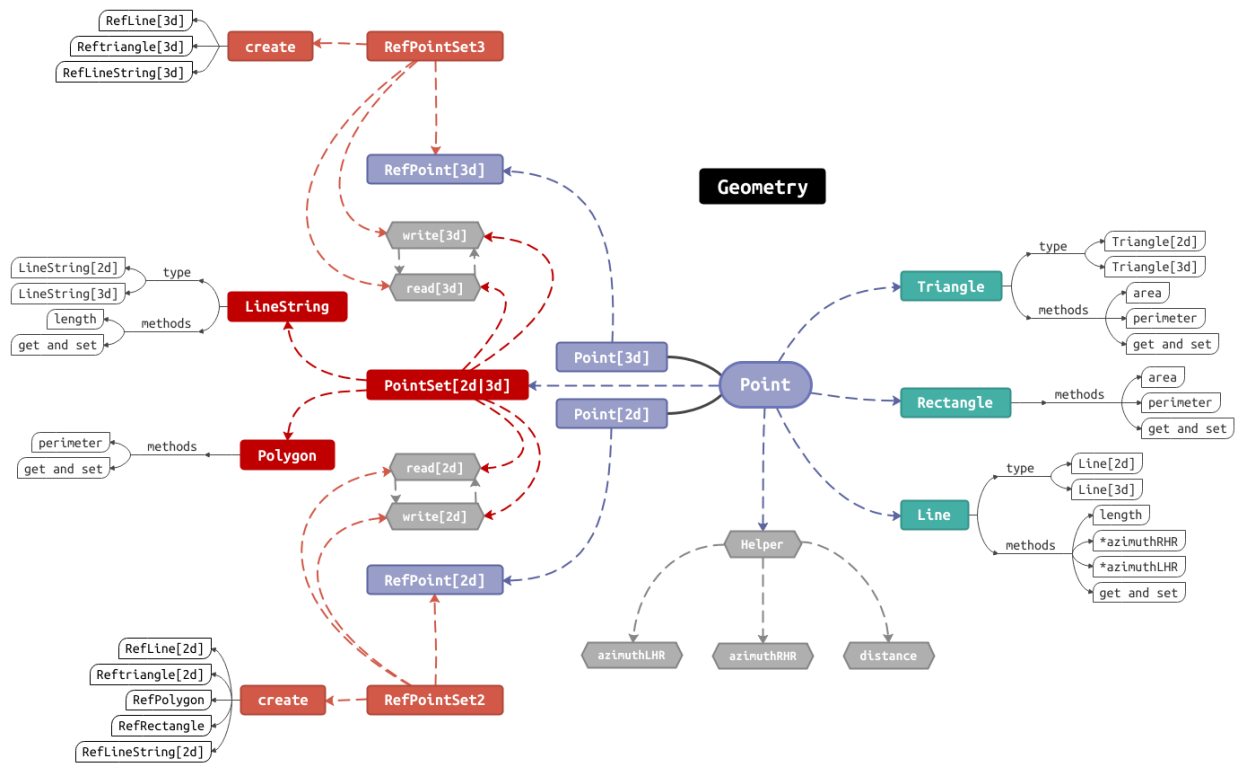
0. Author

`name` csl

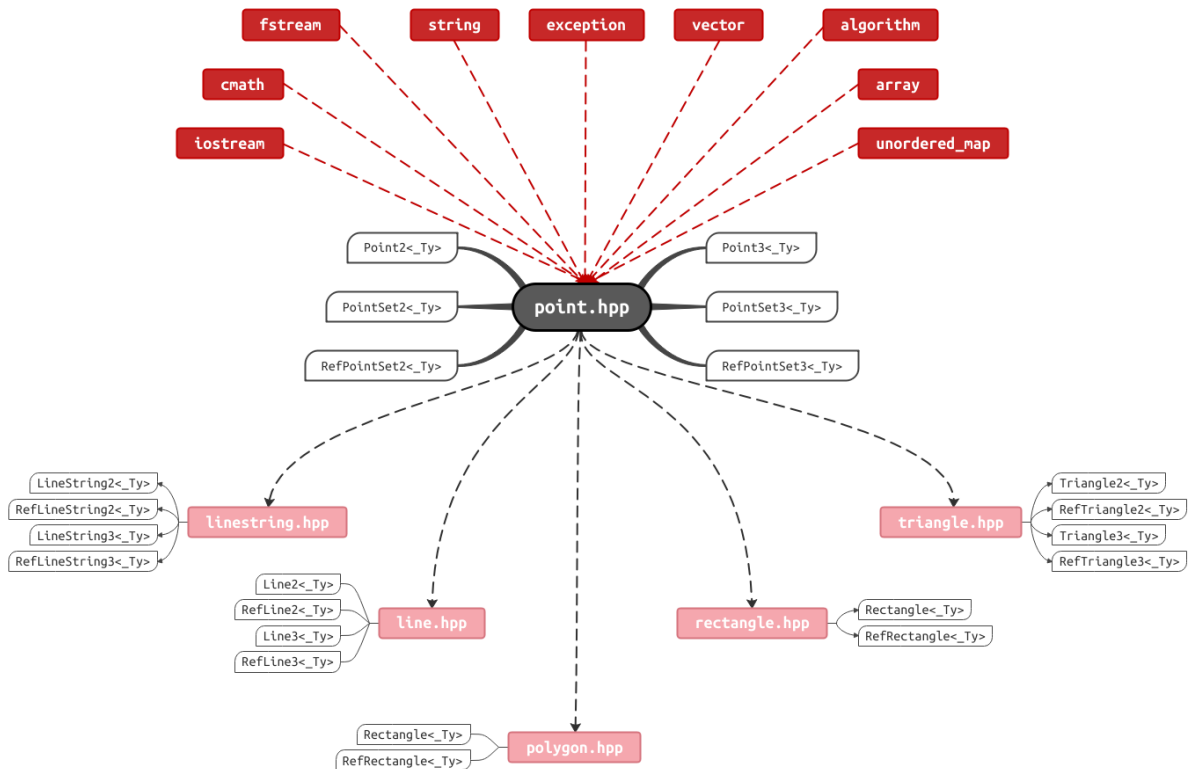`email` 3079625093@qq.com

1. Overview

The CPP library provides two dimension point template classes: `Point2<Ty>` and `Point3<Ty>`. It also provides related operations based on two kinds of points, such as conventional "write" operation, "read" operation and distance calculation of point set, and azimuth calculation based on point2. You can easily use it to assist development. Here are some details of this class.
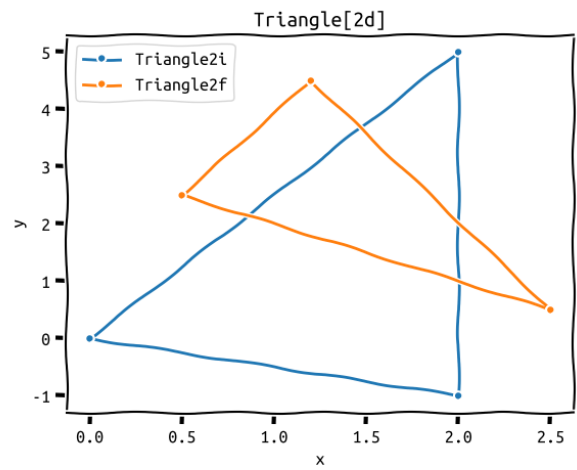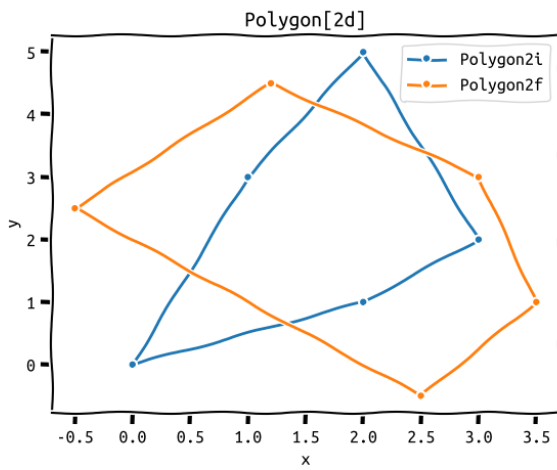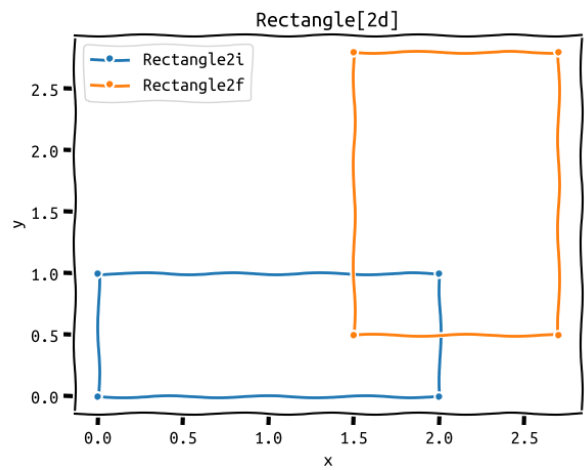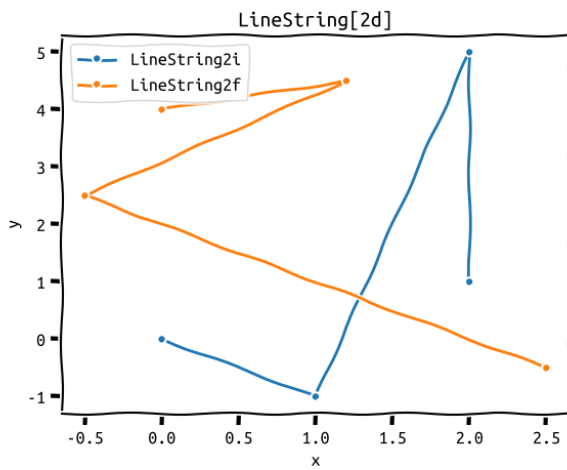And because it's a templat class, you can just copy the head file to your project and use it.

2. Code Structure

Geometry

- RefLine[3d], Reftriangle[3d], RefLineString[3d] → create → RefPointSet3
- RefPoint[3d]
- write[3d], read[3d]
- LineString[2d], LineString[3d] — type; length, get and set — methods → LineString
- PointSet[2d|3d]
- perimeter, get and set — methods → Polygon
- read[2d], write[2d]
- RefPoint[2d]
- RefLine[2d], Reftriangle[2d], RefPolygon, RefRectangle, RefLineString[2d] → create → RefPointSet2
- Point[3d], Point[2d] → Point
- Helper → azimuthLHR, azimuthRHR, distance
- Triangle — type: Triangle[2d], Triangle[3d]; methods: area, perimeter, get and set
- Rectangle — methods: area, perimeter, get and set
- Line — type: Line[2d], Line[3d]; methods: length, *azimuthRHR, *azimuthLHR, get and set

3. Classes Belongs

- fstream, string, exception, vector, algorithm, cmath, array, iostream, unordered_map
- **point.hpp**
  - Point2<_Ty>, Point3<_Ty>
  - PointSet2<_Ty>, PointSet3<_Ty>
  - RefPointSet2<_Ty>, RefPointSet3<_Ty>
- **linestring.hpp** — LineString2<_Ty>, RefLineString2<_Ty>, LineString3<_Ty>, RefLineString3<_Ty>
- **triangle.hpp** — Triangle2<_Ty>, RefTriangle2<_Ty>, Triangle3<_Ty>, RefTriangle3<_Ty>
- **line.hpp** — Line2<_Ty>, RefLine2<_Ty>, Line3<_Ty>, RefLine3<_Ty>
- **rectangle.hpp** — Rectangle<_Ty>, RefRectangle<_Ty>
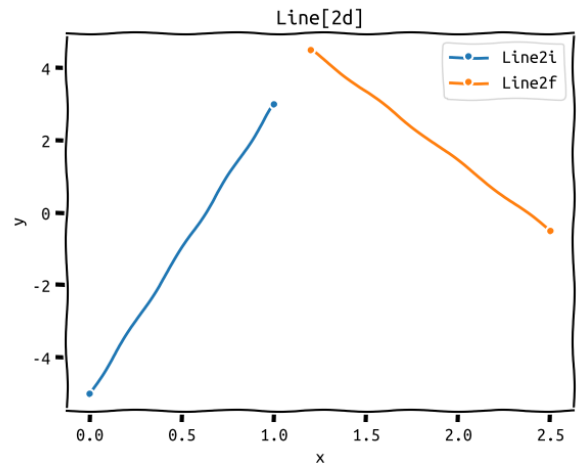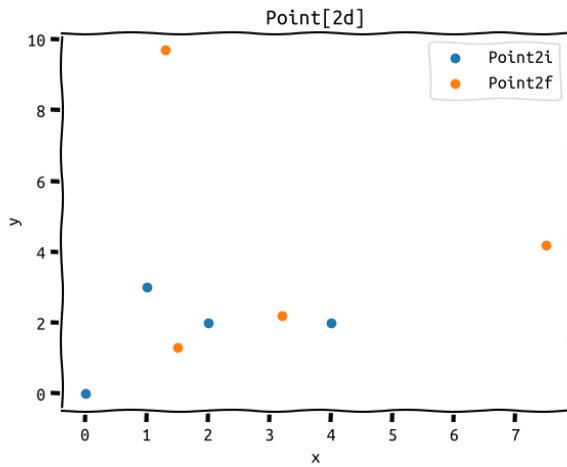- **polygon.hpp** — Rectangle<_Ty>, RefRectangle<_Ty>

4. Figures

5. Using example

Point2<Ty>

```
1   void foo_point2()
2   {
3       PointSet2f ps;
4       ps.push_back({0.6, 0.4});
5       ps.push_back({1.9, 2.7});
6       ps.push_back({0.6, 0.4});
7       ps.push_back({1.9, 2.7});
```

```
 8        try
 9        {
10            // distance between tow points
11            std::cout << distance(ps.front(), ps.back()) << std::endl;
12            // write and read point data
13            // way one.
14            // default write mode : std::ios::out | std::ios::binary
15            ps.write("../output/point2.bin");
16            ps.clear();
17            // default read mode : std::ios::in | std::ios::binary
18            ps.read("../output/point2.bin");
19            // way two.
20            // write mode : std::ios::out
21            ps.write("../output/point2.txt", std::ios::out);
22            ps.clear();
23            // read mode : std::ios::in
24            ps.read("../output/point2.txt", std::ios::in);
25            // print points
26            for (const auto &elem : ps)
27            {
28                std::cout << elem << std::endl;
29            }
30        }
31        catch (const std::exception &e)
32        {
33            std::cerr << e.what() << '\n';
34        }
35        return;
36    }
37    /** output
38     * 2.64197
39     * [0.6, 0.4]
40     * [1.9, 2.7]
41     * [0.6, 0.4]
42     * [1.9, 2.7]
43     */
```

Point3<Ty>

```
 1    void foo_point3()
 2    {
 3        PointSet3f ps;
 4        ps.push_back({0.6, 0.4, 1.1});
 5        ps.push_back({1.9, 2.7, 2.3});
 6        ps.push_back({0.6, 0.4, 1.1});
 7        ps.push_back({1.9, 2.7, 2.3});
 8        try
 9        {
10            // distance between tow points
11            std::cout << distance(ps.front(), ps.back()) << std::endl;
12            // write and read point data
13            // way one.
14            // default write mode : std::ios::out | std::ios::binary
15            ps.write("../output/point3.bin");
16            ps.clear();
17            // default read mode : std::ios::in | std::ios::binary
18            ps.read("../output/point3.bin");
```

```
19
20          // way two.
21          // write mode : std::ios::out
22          ps.write("../output/point3.txt", std::ios::out);
23          ps.clear();
24          // read mode : std::ios::in
25          ps.read("../output/point3.txt", std::ios::in);
26          // print points
27          for (const auto &elem : ps)
28          {
29              std::cout << elem << std::endl;
30          }
31      }
32      catch (const std::exception &e)
33      {
34          std::cerr << e.what() << '\n';
35      }
36      return;
37  }
38  /** output
39   * 2.90172
40   * [0.6, 0.4, 1.1]
41   * [1.9, 2.7, 2.3]
42   * [0.6, 0.4, 3.5]
43   * [1.9, 2.7, 4.6]
44   */
```

PointSet23<Ty>

```
1  void foo_pointset23()
2  {
3      PointSet2f ps;
4      ps.push_back(Point2f(1, 2));
5      ps.push_back(Point2f(2, 3));
6      ps.write("../output/pointset.csv", std::ios::out);
7      ps.clear();
8      ps.read("../output/pointset.csv", std::ios::in);
9      for (const auto &point : ps)
10         std::cout << point << std::endl;
11     return;
12 }
13 /** output
14  * [1, 2]
15  * [2, 3]
16  */
```

Point_cast<Ty>

```
1  void foo_ponitCast_test()
2  {
3      Point3f p(1, 2, 6);
4      Point2f p2(2, 6);
5      auto ary = static_cast<Point3f::ary_type>(p);
```

```
6        auto ary2 = static_cast<Point2f::ary_type>(p2);
7
8        std::cout << ary[0] << ',' << ary[1] << ',' << ary[2] << std::endl;
9        std::cout << ary2[0] << ',' << ary2[1] << std::endl;
10
11       std::cout << Point3f(ary) << std::endl;
12       std::cout << Point2f(ary2) << std::endl;
13
14       return;
15   }
16   /** output
17    * 1,2,6
18    * 2,6
19    * [1, 2, 6]
20    * [2, 6]
21    */
```

Triangle2<Ty>

```
1    void foo_triangle2()
2    {
3        ns_geo::Point2<double> points[3] = {
4            Point2d(0, 0),
5            Point2d(2, 2),
6            Point2d(2, 0)};
7        ns_geo::Triangle2d tri(points);
8        std::cout << tri << std::endl;
9        std::cout << "area : " << tri.area() << std::endl;
10       std::cout << "perimeter : " << tri.perimeter() << std::endl;
11       return;
12   }
13   /** output
14    * {[0, 0], [2, 2], [2, 0]}
15    * area : 2
16    * perimeter : 6.82843
17    */
```

Triangle3<Ty>

```
1    void foo_triangle3()
2    {
3        ns_geo::Point3<double> points[3] = {
4            Point3d(0, 0, 0),
5            Point3d(2, 2, 2),
6            Point3d(2, 0, 0)};
7        ns_geo::Triangle3d tri(points);
8        std::cout << tri << std::endl;
9        std::cout << "area : " << tri.area() << std::endl;
10       std::cout << "perimeter : " << tri.perimeter() << std::endl;
11       return;
12   }
13   /** output
14    * {[0, 0, 0], [2, 2, 2], [2, 0, 0]}
```

```
15   * area : 2.82843
16   * perimeter : 8.29253
17  */
```

Line2<Ty>

```
1  void foo_line2()
2  {
3      ns_geo::Line2d line(Point2d(0, 0), Point2d(2, 2));
4      std::cout << line << std::endl;
5      std::cout << "length : " << line.length() << std::endl;
6      for (const auto &elem : line.points())
7          std::cout << elem << std::endl;
8      return;
9  }
10 /** output
11  * {[0, 0], [2, 2]}
12  * length : 2.82843
13  * [0, 0]
14  * [2, 2]
15  */
```

Line3<Ty>

```
1  void foo_line3()
2  {
3      ns_geo::Line3d line(Point3d(0, 0, 0), Point3d(2, 2, 2));
4      std::cout << line << std::endl;
5      std::cout << "length : " << line.length() << std::endl;
6      for (const auto &elem : line.points())
7          std::cout << elem << std::endl;
8      return;
9  }
10 /** output
11  * {[0, 0, 0], [2, 2, 2]}
12  * length : 3.4641
13  * [0, 0, 0]
14  * [2, 2, 2]
15  */
```

Rectangle<Ty>

```
1  void foo_rectangle()
2  {
3      ns_geo::Rectangled rect(0, 4, 1, 0);
4      std::cout << rect << std::endl;
5      std::cout << "area : " << rect.area() << std::endl;
6      std::cout << "peri : " << rect.perimeter() << std::endl;
```

```
 7        for (const auto &elem : rect.points())
 8            std::cout << elem << std::endl;
 9        return;
10    }
11    /** output
12     * {[0, 4], [1, 0]}
13     * area : 4
14     * peri : 10
15     * [0, 4]
16     * [1, 0]
17     */
```

Polygon<Ty>

```
 1    void foo_polygon()
 2    {
 3        Polygond polygon({Point2d(0, 0),
 4                          Point2d(0, 1),
 5                          Point2d(1, 1),
 6                          Point2d(1, 0)});
 7        std::cout << polygon << std::endl;
 8        std::cout << "perimeter : " << polygon.perimeter() << std::endl;
 9        return;
10    }
11    /** output
12     * {[0, 0], [0, 1], [1, 1], [1, 0]}
13     * perimeter : 4
14     */
```

LineString23<Ty>

```
 1    void foo_lineString23()
 2    {
 3        LineString3d ls({Point3d(0, 0, 9),
 4                         Point3d(0, 1, 9),
 5                         Point3d(1, 1, 9),
 6                         Point3d(1, 0, 9)});
 7        std::cout << ls << std::endl;
 8        std::cout << ls.length() << std::endl;
 9        LineString2d ls2({Point2d(0, 9),
10                          Point2d(1, 9),
11                          Point2d(1, 9),
12                          Point2d(0, 9)});
13        std::cout << ls2 << std::endl;
14        std::cout << ls2.length() << std::endl;
15        return;
16    }
17    /** output
18     * {[0, 0, 9], [0, 1, 9], [1, 1, 9], [1, 0, 9]}
19     * 3
20     * {[0, 9], [1, 9], [1, 9], [0, 9]}
21     * 2
22     */
```

RefPoint23<Ty>

```cpp
void foo_refpoint23()
{
    double ary1[3] = {1, 2, 3};
    RefPoint3d p1(0, RefPoint3d::ary_type{0, 0, 0});
    RefPoint3d p2(1, ary1);
    std::cout << distance(p1, p2) << std::endl;
    std::cout << p1 << std::endl;

    double ary2[2] = {2, 3};
    RefPoint2d p3(0, RefPoint2d::ary_type{0, 0});
    RefPoint2d p4(1, ary2);
    std::cout << distance(p3, p4) << std::endl;
    std::cout << p3 << std::endl;
}
/** output
 * 3.74166
 * {0: [0, 0, 0]}
 * 3.60555
 * {0: [0, 0]}
 */
```

RefPointSet23<Ty>

```cpp
void foo_refpointset23()
{
    double ary2[2] = {2, 3};
    RefPointSet2d rps2;
    rps2.insert({0, RefPoint2d::ary_type{0, 0}});
    rps2.insert({1, ary2});
    rps2.insert({2, RefPoint2d::ary_type{0, 0}});
    rps2.insert({4, ary2});
    for (const auto &refp : rps2)
        std::cout << refp.second << std::endl;
    std::cout << rps2.size() << std::endl;

    RefPointSet3d rps3;
    rps3.insert({0, RefPoint3d::ary_type{0, 0, 0}});
    rps3.insert({1, RefPoint3d::ary_type{0, 1, 0}});
    rps3.insert({2, RefPoint3d::ary_type{0, 0, 1}});
    rps3.insert({3, RefPoint3d::ary_type{1, 0, 0}});
    for (const auto &refp : rps3)
        std::cout << refp.second << std::endl;
    std::cout << rps3.size() << std::endl;
}
/** output
 * {0: [0, 0]}
 * {2: [0, 0]}
 * {4: [2, 3]}
 * {1: [2, 3]}
 * 4
```

```
28    * {4: [1, 0, 0]}
29    * {2: [0, 0, 1]}
30    * {1: [0, 1, 0]}
31    * {0: [0, 0, 0]}
32    * 4
33    */
```

RefLine23<Ty>

```
1   void foo_refline2()
2   {
3       double ary2[2] = {2, 3};
4       RefPointSet2d rps;
5       rps.insert({0, RefPoint2d::ary_type{0, 0}});
6       rps.insert({1, ary2});
7       rps.insert({2, RefPoint2d::ary_type{0, 0}});
8       rps.insert({4, ary2});
9       for (const auto &refp : ps)
10          std::cout << refp.second << std::endl;
11
12      auto refline = rps.createRefLine2(0, 1);
13      std::cout << refline << std::endl;
14      std::cout << refline.length() << std::endl;
15  }
16  /** output
17   * {0: [0, 0]}
18   * {2: [0, 0]}
19   * {4: [2, 3]}
20   * {1: [2, 3]}
21   * {0: [0, 0], 1: [2, 3]}
22   * 3.60555
23   */
24
25  void foo_refline3()
26  {
27      RefPointSet3d rps;
28      rps.insert({0, RefPoint3d::ary_type{0, 0, 0}});
29      rps.insert({1, RefPoint3d::ary_type{0, 1, 0}});
30      rps.insert({2, RefPoint3d::ary_type{0, 0, 1}});
31      rps.insert({3, RefPoint3d::ary_type{1, 0, 0}});
32      for (const auto &refp : ps)
33          std::cout << refp.second << std::endl;
34
35      auto refline = rps.createRefLine3(0, 1);
36      std::cout << refline << std::endl;
37      std::cout << refline.length() << std::endl;
38      auto ary = refline.points();
39  }
40  /** output
41   * {0: [0, 0, 0]}
42   * {2: [0, 0, 1]}
43   * {4: [1, 0, 0]}
44   * {1: [0, 1, 0]}
45   * {0: [0, 0, 0], 1: [0, 1, 0]}
46   * 1
47   */
```

RefRectangle<Ty>

```cpp
void foo_refrectangle()
{
    double ary2[2] = {2, 3};
    RefPointSet2d rps;
    rps.insert({0, RefPoint2d::ary_type{0, 0}});
    rps.insert({1, ary2});
    rps.insert({2, RefPoint2d::ary_type{0, 0}});
    rps.insert({4, ary2});
    for (const auto &refp : rps)
        std::cout << refp.second << std::endl;

    auto rect = rps.createRefRectangle(0, 1);
    std::cout << rect << std::endl;
    std::cout << rect.area() << std::endl;
    std::cout << rect.perimeter() << std::endl;
}
/** output
 * {0: [0, 0]}
 * {2: [0, 0]}
 * {4: [2, 3]}
 * {1: [2, 3]}
 * {0: [0, 0], 1: [2, 3]}
 * 6
 * 10
 */
```

RefTriangle23<Ty>

```cpp
void foo_reftriangle2()
{
    double ary2[2] = {2, 3};
    RefPointSet2d rps;
    rps.insert({0, RefPoint2d::ary_type{0, 0}});
    rps.insert({1, ary2});
    rps.insert({2, RefPoint2d::ary_type{0, 0}});
    rps.insert({4, ary2});
    for (const auto &refp : rps)
        std::cout << refp.second << std::endl;
    auto tri = rps.createRefTriangle2(0, 1, 2);

    std::cout << tri << std::endl;
    std::cout << tri.perimeter() << std::endl;
    std::cout << tri.area() << std::endl;
}
/** output
 * {0: [0, 0]}
 * {2: [0, 2]}
 * {4: [3, 0]}
 * {1: [1, 0]}
 * {0: [0, 0], 1: [1, 0], 2: [0, 2]}
 * 5.23607
 * 1
 */
```

```
27  void foo_reftriangle3()
28  {
29      RefPointSet3d rps;
30      rps.insert({0, RefPoint3d::ary_type{0, 0, 0}});
31      rps.insert({1, RefPoint3d::ary_type{0, 1, 0}});
32      rps.insert({2, RefPoint3d::ary_type{0, 0, 1}});
33      rps.insert({3, RefPoint3d::ary_type{1, 0, 0}});
34      for (const auto &refp : rps)
35          std::cout << refp.second << std::endl;
36
37      auto tri = rps.createRefTriangle3(0, 1, 2);
38      std::cout << tri << std::endl;
39      std::cout << tri.area() << std::endl;
40      std::cout << tri.perimeter() << std::endl;
41  }
42  /** output
43   * {0: [0, 0, 0]}
44   * {2: [0, 0, 1]}
45   * {4: [1, 0, 0]}
46   * {1: [0, 1, 0]}
47   * {0: [0, 0, 0], 1: [0, 1, 0], 2: [0, 0, 1]}
48   * 0.5
49   * 3.41421
50   */
```

RefPolygon<Ty>

```
1   void foo_refpolygon()
2   {
3       RefPointSet2d rps;
4       rps.insert({0, RefPoint2d::ary_type{0, 0}});
5       rps.insert({1, RefPoint2d::ary_type{1, 0}});
6       rps.insert({2, RefPoint2d::ary_type{1, 1}});
7       rps.insert({4, RefPoint2d::ary_type{0, 1}});
8
9       auto polygon = rps.createRefPolygon({0, 1, 2, 4});
10      std::cout << polygon << std::endl;
11      std::cout << "perimeter : " << polygon.perimeter() << std::endl;
12  }
13  /** output
14   * {0: [0, 0], 1: [1, 0], 2: [1, 1], 4: [0, 1]}
15   * perimeter : 4
16   */
```

RefLinestring23<Ty>

```
1   void foo_reflinestring2()
2   {
3       RefPointSet2d rps;
4       rps.insert({0, RefPoint2d::ary_type{0, 0}});
5       rps.insert({1, RefPoint2d::ary_type{1, 0}});
6       rps.insert({2, RefPoint2d::ary_type{1, 1}});
7       rps.insert({4, RefPoint2d::ary_type{0, 1}});
```

```
8       auto ls = rps.createRefLineString2({0, 1, 2, 4});
9       std::cout << ls << std::endl;
10      std::cout << "length : " << ls.length() << std::endl;
11  }
12  /** output
13   * {0: [0, 0], 1: [1, 0], 2: [1, 1], 4: [0, 1]}
14   * length : 3
15   */
16
17
18  void foo_reflinestring3()
19  {
20      RefPointSet3d rps;
21      rps.insert({0, RefPoint3d::ary_type{0, 0, 0}});
22      rps.insert({1, RefPoint3d::ary_type{0, 1, 0}});
23      rps.insert({2, RefPoint3d::ary_type{0, 0, 1}});
24      rps.insert({3, RefPoint3d::ary_type{1, 0, 0}});
25
26      auto ls = rps.createRefLineString3({0, 1, 2, 4});
27      std::cout << ls << std::endl;
28      std::cout << "length : " << ls.length() << std::endl;
29  }
30   /** output
31    * {0: [0, 0, 0], 1: [0, 1, 0], 2: [0, 0, 1], 4: [1, 0, 0]}
32    * length : 3.82843
33    */
```

RefPointSet_WriteRead23<Ty>

```
1   void foo_refpointset2_write()
2   {
3       RefPointSet2d rps;
4       rps.insert({0, RefPoint2d::ary_type{0, 0}});
5       rps.insert({1, RefPoint2d::ary_type{1, 0}});
6       rps.insert({2, RefPoint2d::ary_type{1, 1}});
7       rps.insert({4, RefPoint2d::ary_type{0, 1}});
8
9       rps.write("../output/refpointset2.bin");
10      rps.clear();
11      rps.read("../output/refpointset2.bin");
12      for (const auto &[id, refp] : rps)
13          std::cout << refp << std::endl;
14  }
15  /** output
16   * {1: [1, 0]}
17   * {4: [0, 1]}
18   * {2: [1, 1]}
19   * {0: [0, 0]}
20   */
21  void foo_refpointset3_write()
22  {
23      RefPointSet3d rps;
24      rps.insert({0, RefPoint3d::ary_type{0, 0, 0}});
25      rps.insert({1, RefPoint3d::ary_type{0, 1, 0}});
26      rps.insert({2, RefPoint3d::ary_type{0, 0, 1}});
27      rps.insert({3, RefPoint3d::ary_type{1, 0, 0}});
28
```

```
29      rps.write("../output/refpointset3.bin");
30      rps.clear();
31      rps.read("../output/refpointset3.bin");
32      for (const auto &[id, refp] : rps)
33          std::cout << refp << std::endl;
34  }
35  /** output
36   * {1: [0, 1, 0]}
37   * {4: [1, 0, 0]}
38   * {2: [0, 0, 1]}
39   * {0: [0, 0, 0]}
40   */
```

For other implementation details, please refer to the source code.