

```
1 #include "Matrix3x3_PS1.h"
2
3 Matrix3x3 Matrix3x3::operator*(const Matrix3x3& aOther) const noexcept {
4     return Matrix3x3(
5         Vector3D(row(0).dot(aOther.column(0)), row(0).dot(aOther.column(1)), row(0).dot(aOther.column(2))),
6         Vector3D(row(1).dot(aOther.column(0)), row(1).dot(aOther.column(1)), row(1).dot(aOther.column(2))),
7         Vector3D(row(2).dot(aOther.column(0)), row(2).dot(aOther.column(1)), row(2).dot(aOther.column(2)))
8     );
9 }
10
11 std::ostream& operator<<(std::ostream& aOStream, const Matrix3x3& aMatrix)
12 {
13     size_t lIndex = 0;
14     aOStream << "[";
15     while (lIndex < 3) {
16         aOStream << aMatrix.row(lIndex).toString();
17         if(lIndex++ != 2) aOStream << ",";
18     }
19     aOStream << "]";
20     return aOStream;
21 }
22
23 float Matrix3x3::det() const noexcept {
24     const Vector3D& row0 = row(0);
25     const Vector3D& row1 = row(1);
26     const Vector3D& row2 = row(2);
27
28     return row0.x() * (row1.y() * row2.w() - row1.w() * row2.y())
29         - row0.y() * (row1.x() * row2.w() - row1.w() * row2.x())
30         + row0.w() * (row1.x() * row2.y() - row1.y() * row2.x());
31 }
32
33 bool Matrix3x3::hasInverse() const noexcept {
34     return det() != 0.0f;
35 }
36
37 Matrix3x3 Matrix3x3::transpose() const noexcept {
38     return Matrix3x3(
39         Vector3D(column(0)[0], column(0)[1], column(0)[2]),
40         Vector3D(column(1)[0], column(1)[1], column(1)[2]),
41         Vector3D(column(2)[0], column(2)[1], column(2)[2])
42     );
43 }
44
45 Matrix3x3 Matrix3x3::inverse() const noexcept {
```

```
46
47     if (det() != 0.0f) {
48         float fInverseDetM = 1.0f / det();
49
50         const Vector3D& row0 = row(0);
51         const Vector3D& row1 = row(1);
52         const Vector3D& row2 = row(2);
53
54         float fInverseElement00 = (row1.y() * row2.w() - row1.w() * row2.y() * fInverseDetM;
55         float fInverseElement01 = -(row0.y() * row2.w() - row0.w() * row2.y() * fInverseDetM;
56         float fInverseElement02 = (row0.y() * row1.w() - row0.w() * row1.y() * fInverseDetM;
57
58         float fInverseElement10 = -(row1.x() * row2.w() - row1.w() * row2.x() * fInverseDetM;
59         float fInverseElement11 = (row0.x() * row2.w() - row0.w() * row2.x() * fInverseDetM;
60         float fInverseElement12 = -(row0.x() * row1.w() - row0.w() * row1.x() * fInverseDetM;
61
62         float fInverseElement20 = (row1.x() * row2.y() - row1.y() * row2.x() * fInverseDetM;
63         float fInverseElement21 = -(row0.x() * row2.y() - row0.y() * row2.x() * fInverseDetM;
64         float fInverseElement22 = (row0.x() * row1.y() - row0.y() * row1.x() * fInverseDetM;
65
66         return Matrix3x3(Vector3D(fInverseElement00, fInverseElement01, fInverseElement02),
67                           Vector3D(fInverseElement10, fInverseElement11, fInverseElement12),
68                           Vector3D(fInverseElement20, fInverseElement21, fInverseElement22));
69     }
70     else {
71         std::cout << "Determinate of M is Zero";
72     }
73 }
74
75
76
77
78
79
80
81
82
```