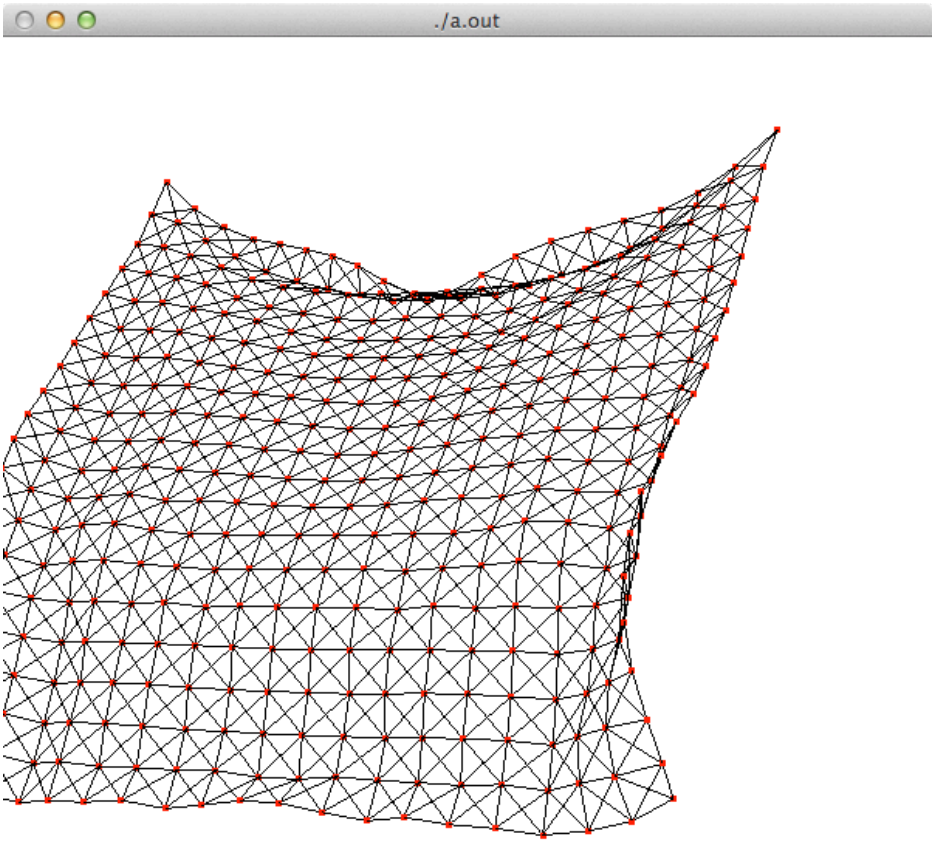
```
// ベクトル演算の定義(クラスの外(トップレベル)に記述)
inline Vector3d operator*( const double& k, const Vector3d& v ) { return Vector3d( k*v.x, k*v.y, k*v.z ); }
inline Vector3d operator*( const Vector3d& v, const double& k ) { return Vector3d( v.x*k, v.y*k, v.z*k ); }
inline Vector3d operator/( const Vector3d& v, const double& k ) { return Vector3d( v.x/k, v.y/k, v.z/k ); }

void updateCloth(void) {
    // ★次の手順で質点の位置を決定する
    // 1. 質点に働く力を求める

    // 1-1. 質点に働く力をリセット
    for(int y = 0; y < POINT_NUM; y++) {
        for(int x = 0; x < POINT_NUM; x++) {
            cloth->points[x][y].f.set(0,0,0);
        }
    }

    // 1-2. バネの両端の質点に力を働かせる
    for(int i = 0; i < cloth->springs.size(); i++) {
        Spring *spring = cloth->springs[i];
        double restLength = spring->restLength;
        double nowLength = (spring->p0->p - spring->p1->p).length();
        Vector3d ep01 = (spring->p1->p - spring->p0->p);
        Vector3d ep10 = (spring->p0->p - spring->p1->p);
        ep01.normalize();
        ep10.normalize();
        spring->p0->f += Ks * (nowLength - restLength) * ep01;
        spring->p1->f += Ks * (nowLength - restLength) * ep10;
    }

    // 1-3. 重力、空気抵抗による力を加算する
    for(int y = 0; y < POINT_NUM; y++) {
        for(int x = 0; x < POINT_NUM; x++) {
            cloth->points[x][y].f += Mass * gravity - Dk * cloth->points[x][y].v;
        }
    }

    // bFixed = true の点に働く力(合力だし)を0とする。
    for(int y = 0; y < POINT_NUM; y++) {
        for(int x = 0; x < POINT_NUM; x++) {
            if(cloth->points[x][y].bFixed == true) {
                cloth->points[x][y].f.set(0,0,0);
            }
        }
    }

    // 2., 3., 4. 加速度・速度・位置を求める
    for(int y = 0; y < POINT_NUM; y++) {
        for(int x = 0; x < POINT_NUM; x++) {
            // 2. 質点の加速度を求める
            Vector3d a = cloth->points[x][y].f / Mass;
            // 3. 質点の速度を更新する
            cloth->points[x][y].v = cloth->points[x][y].v + dT * a;
            // 4. 質点の位置を更新する
            cloth->points[x][y].p = cloth->points[x][y].p + dT * cloth->points[x][y].v;
        }
    }
}
```



Vector3dクラスの中に以下のコード

```
inline Vector3d operator*( const double& k, const Vector3d& v ) { return Vector3d( k*v.x, k*v.y, k*v.z ); }
inline Vector3d operator*( const Vector3d& v, const double& k ) { return Vector3d( v.x*k, v.y*k, v.z*k ); }
inline Vector3d operator/( const Vector3d& v, const double& k ) { return Vector3d( v.x/k, v.y/k, v.z/k ); }
```

を書いたら、+は3項演算子にはできないとエラーが出た。クラスの中に書くのと外に書くのでは違うらしい。

最初は、それぞれのクラスがどういう構造になっていて、何を取得して何を代入するのか、アニメーションさせるのはどうやってやるのか(アニメーションは既に用意されていて何も描く必要はなかった)と戸惑った。最終的にできあがった物理演算は簡潔なコードになって意外な感じがする。