代码使用说明

文件说明：

- main.cpp ： 测试 MeshHelper 库所用的 main 函数测试文件。
- mesh.h ： MeshHelper 库定义文件。
- mesh.cpp ： MeshHelper 库实现文件。
- Makefile ： 项目的 Makefile
- testmesh.m ： 测试所用的 mesh file。

main 说明：

用户自定义的 kernel 函数，后续用于传入 Loop：

```
int myKernelFun(Vertex *v) {
    double sum = *(double*)v->info;
    for (auto u : v->adj) {
        sum += *(double*)u->info;
    }
    *(double*)(v->info) = sum / (v->adj.size() + 1);
    return 0;
}
```

测试功能一：模拟仿真核运算

```
/* TEST 1 */
/* Example 1: Kernel Function */
MeshHelper *mh = new MeshHelper();
mh->LoadMesh("testmesh.m");
const Vertex *vertices = mh->GetVertices();
int n = mh->GetNumVer();
for (int i = 0; i < n; ++i) {
    double *p = new double;
    *p = vertices[i].x * 100.0;
    mh->SetInfo(i, p);
}

printf("------------original mesh data--------------------\n");
mh->PrintInfo();

printf("------------start loop----------------------------\n");
int loop_num = 10;
for(int i = 0; i < loop_num; i++) {
    mh->Loop(myKernelFun, sizeof(double));
    if(i == 5) {
        printf("---track data after loop %d----\n", i);
        mh->PrintInfo();
        printf("-----------------------------\n");
    }
}
printf("------------end loop------------------------------\n");

printf("------------after loop by kernel function----------\n");
mh->PrintInfo();
```

开始Loop

自定义哪个时间点输出中间信息

测试功能二：生成自己的 Mesh 文件

```
/* TEST 2 */
/* Example 2: Generate New Mesh */
MeshHelper *nh = new MeshHelper();
Vertex *verts = new Vertex[5];
for (int i = 0; i < 4; ++i) {
    verts[i].x = (i & 1) * 10.0;
    verts[i].y = (i >> 1 & 1) * 10.0;
}
verts[4].x = 5.0;
verts[4].y = 5.0;
int *edges = new int[16];
int cur = 0, nex;
for (int i = 0; i < 4; ++i) {
    nex = cur ^ ((i & 1) + 1);
    edges[i * 2] = cur;
    edges[i * 2 + 1] = nex;
    cur = nex;
}
for (int i = 0; i < 4; ++i) {
    edges[(i + 4) * 2] = i;
    edges[(i + 4) * 2 + 1] = 4;
}
nh->Generate(verts, 5, edges, 8);
nh->Save("mymesh.m");
```

Save

编译-测试说明：

make 可以编译出 libmesh.so 动态库，供开发者按需使用。

make example 可以编译出带 main 函数的例子程序，并运行查看测试结果。

make clean 清理项目。

```
zzm@zzm-VirtualBox:/media/sf_dblinuxshare/project$ make example
g++ -Wall -std=c++11 -fPIC -g -c mesh.cpp
g++ -Wall -std=c++11 -fPIC -g mesh.o -shared -o libmesh.so
g++ -Wall -std=c++11 -fPIC -g -c main.cpp
g++ -Wall -std=c++11 -fPIC -g main.o -L. -lmesh -Wl,-rpath=. -o main
./main
```

```
------------original mesh data---------------------
Number of vertices = 11
Number of edges = 22
Vertices info:
0: position : (0.000000, 0.000000) adj vertex:  10 7 1 info:  0.000000
1: position : (0.100000, 0.000000) adj vertex:  10 5 0 info:  10.000000
2: position : (0.100000, 0.300000) adj vertex:  9 3 4 info:  10.000000
3: position : (0.000000, 0.300000) adj vertex:  9 6 2 info:  0.000000
4: position : (0.100000, 0.200000) adj vertex:  9 8 5 2 info:  10.000000
5: position : (0.100000, 0.100000) adj vertex:  7 8 10 1 4 info:  10.000000
6: position : (0.000000, 0.200000) adj vertex:  9 8 7 3 info:  0.000000
7: position : (0.000000, 0.100000) adj vertex:  5 8 10 0 6 info:  0.000000
8: position : (0.050000, 0.150000) adj vertex:  9 7 6 4 5 info:  5.000000
9: position : (0.050000, 0.250000) adj vertex:  4 8 6 3 2 info:  5.000000
10: position : (0.050000, 0.050000) adj vertex:  0 7 5 1 info:  5.000000
------------start loop----------------------------
---track data after loop 5----
Number of vertices = 11
Number of edges = 22
Vertices info:
0: position : (0.000000, 0.000000) adj vertex:  10 7 1 info:  4.988132
1: position : (0.100000, 0.000000) adj vertex:  10 5 0 info:  5.011868
2: position : (0.100000, 0.300000) adj vertex:  9 3 4 info:  5.023391
3: position : (0.000000, 0.300000) adj vertex:  9 6 2 info:  4.976609
4: position : (0.100000, 0.200000) adj vertex:  9 8 5 2 info:  5.039116
5: position : (0.100000, 0.100000) adj vertex:  7 8 10 1 4 info:  5.020493
6: position : (0.000000, 0.200000) adj vertex:  9 8 7 3 info:  4.960884
7: position : (0.000000, 0.100000) adj vertex:  5 8 10 0 6 info:  4.979507
8: position : (0.050000, 0.150000) adj vertex:  9 7 6 4 5 info:  5.000000
9: position : (0.050000, 0.250000) adj vertex:  4 8 6 3 2 info:  5.000000
10: position : (0.050000, 0.050000) adj vertex:  0 7 5 1 info:  5.000000
-----------------------------
------------end loop------------------------------
------------after loop by kernel function----------
Number of vertices = 11
Number of edges = 22
Vertices info:
0: position : (0.000000, 0.000000) adj vertex:  10 7 1 info:  4.999621
1: position : (0.100000, 0.000000) adj vertex:  10 5 0 info:  5.000379
2: position : (0.100000, 0.300000) adj vertex:  9 3 4 info:  5.000737
3: position : (0.000000, 0.300000) adj vertex:  9 6 2 info:  4.999263
4: position : (0.100000, 0.200000) adj vertex:  9 8 5 2 info:  5.001242
5: position : (0.100000, 0.100000) adj vertex:  7 8 10 1 4 info:  5.000642
6: position : (0.000000, 0.200000) adj vertex:  9 8 7 3 info:  4.998758
7: position : (0.000000, 0.100000) adj vertex:  5 8 10 0 6 info:  4.999358
8: position : (0.050000, 0.150000) adj vertex:  9 7 6 4 5 info:  5.000000
9: position : (0.050000, 0.250000) adj vertex:  4 8 6 3 2 info:  5.000000
10: position : (0.050000, 0.050000) adj vertex:  0 7 5 1 info:  5.000000
```