碰撞检测 API-演示程序

唐敏（浙江大学 tang_m@zju.edu.cn）
2020/11/3

1. 目录结构

physika-cd-v3	碰撞检测库源代码+Ubuntu Makefile
obj-cd-physika	调用碰撞检测库的演示程序（Visual Studio 2015）
data	本文档中使用的两个 obj 文件，可以做为演示程序是输入

2. 编译环境

CUDA 9.2/10.2/11.0+GCC/Visual Studio 2015+GLUT。

3. 模块功能

在 GPU 上计算 Mesh 之间的碰撞情况，返回发生碰撞的三角形对。

4. 运行

objCD.exe cloth.obj body.obj

检测命题行提供的两个 obj 文件中三角形网格之间的碰撞情况。

程序运行后，会看到：

使用鼠标左键，SHIFT+鼠标左键，CTRL+鼠标左键，可以实现观察的旋转、平移、缩放功能。

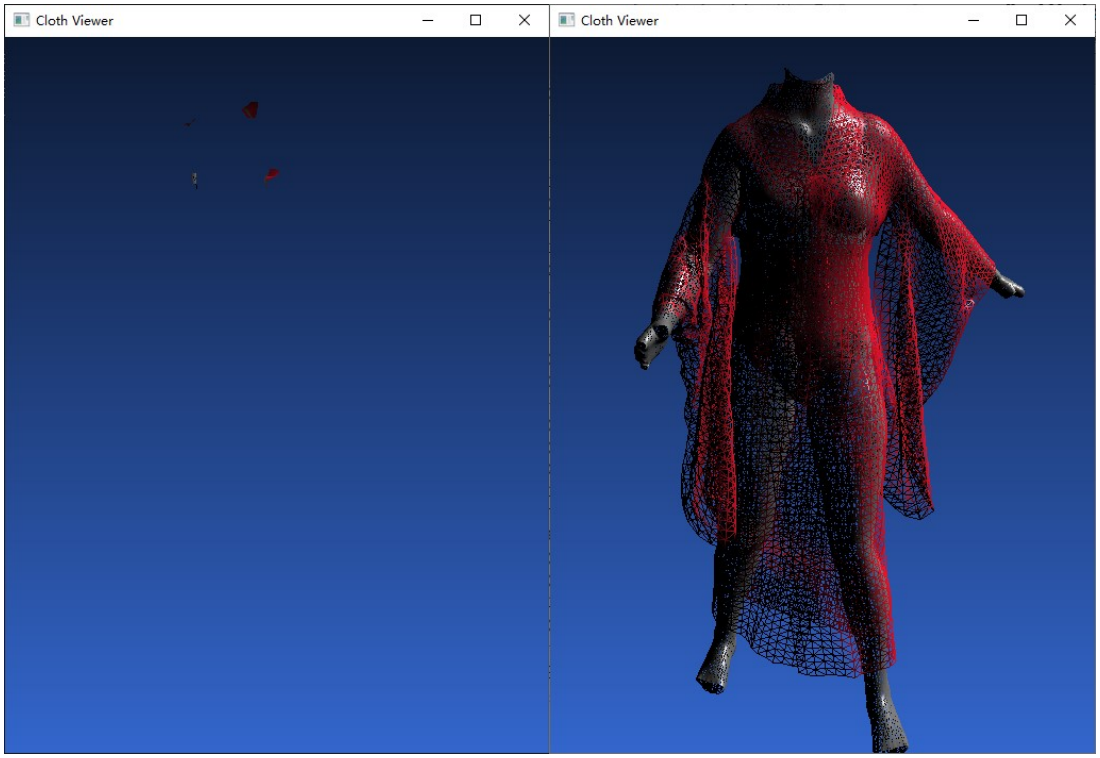
键盘按‘2’，会检测衣服和人体之间的碰撞情况，输入如下：

```
Read obj file don (#tri=14578, #vtx=16556)
g_pair.init: GPU memory usage: used = 1371.936719, free = 4772.063281 MB, total = 6144.000000 MB
g_pair.init: GPU memory usage: used = 1677.124219, free = 4466.875781 MB, total = 6144.000000 MB
g_pair.init: GPU memory usage: used = 1829.936719, free = 4314.063281 MB, total = 6144.000000 MB
g_front.init: GPU memory usage: used = 2674.186719, free = 3469.813281 MB, total = 6144.000000 MB
676770
Before propogate, length = 676770
After propogate, length = 676770
pair = 60168
collision num = 63
1: (0, 5034) - (1, 7863)
2: (0, 16957) - (1, 12726)
3: (0, 17402) - (1, 8806)
4: (0, 11858) - (1, 12640)
```

```
5: (0, 11858) - (1, 12641)
6: (0, 16955) - (1, 12640)
7: (0, 16955) - (1, 12632)
8: (0, 17523) - (1, 9652)
9: (0, 17519) - (1, 9652)
10: (0, 5033) - (1, 7863)
11: (0, 16957) - (1, 12641)
12: (0, 17400) - (1, 8806)
13: (0, 11856) - (1, 12727)
14: (0, 11856) - (1, 12641)
15: (0, 11858) - (1, 12632)
16: (0, 5032) - (1, 7863)
17: (0, 17403) - (1, 8806)
18: (0, 5034) - (1, 8887)
19: (0, 17402) - (1, 8808)
20: (0, 17434) - (1, 9652)
21: (0, 5033) - (1, 8887)
22: (0, 17523) - (1, 9663)
23: (0, 17520) - (1, 9652)
24: (0, 16954) - (1, 12640)
25: (0, 11857) - (1, 12641)
26: (0, 16957) - (1, 12727)
27: (0, 11856) - (1, 12726)
28: (0, 17270) - (1, 9663)
29: (0, 17435) - (1, 9652)
30: (0, 16954) - (1, 12641)
31: (0, 17270) - (1, 9652)
32: (0, 5032) - (1, 8887)
Found 32 inter-object contacts...
```

一共找到 32 个碰撞。

可以通过按键‘t’，把碰撞的三角形显示出来，或者关闭显。按键‘w’可以切换线框/填充显示。



放大后可以清楚看到碰撞穿透情况：

5. API 调用说明

API 函数参见 Collision.h，具体调用见 cmodel.cpp：

```
#include "Collision.h"
```

```
Collision cdMgr;
```

在装入人体和服装 obj 模型时，调用 API 函数：

```
void initObj(const char *ofile)
{
	unsigned int numVtx = 0, numTri = 0;
	vec3f *vtxs = NULL;
	tri3f *tris = NULL;
	vec2f *texs = NULL;
	tri3f *ttris = NULL;

	double scale = 1.;
	vec3f shift(0, 0, 0);

	if (readobjfile(ofile, numVtx, numTri, tris, vtxs, scale, shift, false)) {
		//printf("loading %s ...\n", buff);

		lions[0] = new mesh(numVtx, numTri, tris, vtxs);
		lions[0]->updateNrms();
		printf("Read obj file don (#tri=%d, #vtx=%d)\n", numTri, numVtx);

		vector<unsigned int> ttris;
		vector<vec3f> tvtxs;

		for (int i = 0; i < numTri; i++) {
			ttris.push_back(tris[i].id0());
			ttris.push_back(tris[i].id1());
			ttris.push_back(tris[i].id2());
		}
		for (int i = 0; i < numVtx; i++)
			tvtxs.push_back(vtxs[i]);

		cdMgr.Transform_Mesh(numVtx, numTri, ttris, tvtxs, tvtxs, 1);
	}
}
```

按键‘2‘后，进行碰撞检测：

```
void checkModel()
{
	cdMgr.Collid();
```

```
    int ret = cdMgr.getNumContacts();
    //printf("Found %d contacts...\n");

    vector<vector<tri_pair> > pairs = cdMgr.getContactPairs();

    int count = 0;
    for (int i = 0; i < pairs.size(); i++) {
        tri_pair &t1 = pairs[i][0];
        tri_pair &t2 = pairs[i][1];

        if (t1.id0() == t2.id0()) //self cd
            continue;

        printf("%d: (%d, %d) - (%d, %d)\n", count + 1, t1.id0(), t1.id1(), t2.id0(), t2.id1());
        addTri(t1.id0(), t1.id1());
        addTri(t2.id0(), t2.id1());
        count++;
    }

    printf("Found %d inter-object contacts...\n", count);
}
```

最后收集的碰撞三角形会在显示模块绘制出来（按键't'以后）。

6. API 调用方

Obj-viewer.cpp 中使用了 GLUT 进行三维模型的显示，并通过键盘/鼠标回调函数进行显示/碰撞检测。