有很多 opengl 的绘制部分，比如 camera.h,shader.h，不需要太关注。

初始化部分：

```
85    Model teapot0 = Model("../model/teapot.obj");
86    Model teapot1 = Model("../model/teapot.obj");
87
88    // tell opengl for each sampler to which texture unit it belongs to (only has to be done once)
89    // -------------------------------------------------------------------------------------------
90    ourShader.use();
91    teapot1.update(-2.3);
```

Model 读入 obj 文件，存储顶点信息和三角面片信息

Model 的 update 函数：

```
111  void Model::update(double dt) {
112      for (int i = 0; i < verts_.size(); i++) {
113          verts_[i] += vec3d(0, 1.0, 0)*dt;
114      }
115  }
```

沿着 y 轴正方向移动 1.0*dt 的距离。

While 循环里的步骤

```
102          teapot1.update(0.001);
103          CollidePair.clear();
104          CollideDetection::execute(&teapot0, &teapot1, &CollidePair);
105          printf("collide pair num : %d\n", CollidePair.size());
```

首先 update，也就是模型移动一下。
然后解释一下 CollidePair

```
6   class BVHIndexPair
7   {
8   public:
9
10      BVHIndexPair(const unsigned int& id1, const unsigned int& id2);
11
12      void getTriPair(unsigned int* id1, unsigned int* id2) const;
13
14      void sort();
15
16      bool operator<(const BVHIndexPair& other) const;
17
18  private:
19
20      unsigned int m_id[2];
21  };
```

它里面存了两个 unsigned int ，也就是存放两个三角形的面片的序号，比如我的代码中，teapot0 的第一个三角形和 teapot1 的第三个三角形检测到碰撞，
那么 m_id[0] = 1,m_id[1]=3。

在 while 循环中，首先清除一下上一帧的碰撞检测的结果，然后 execute 碰撞检测。

```
void CollideDetection::execute(Model* model0, Model* model1, std::vector<BVHIndexPair>* f) {
    //还原颜色
    for (int i = 0; i < model0->colors_.size(); i++) {
        model0->colors_[i] = vec3f(1.0f, 1.0f, 1.0f);
    }

    for (int i = 0; i < model1->colors_.size(); i++) {
        model1->colors_[i] = vec3f(1.0f, 1.0f, 1.0f);
    }

    std::vector<BVHIndexPair> BoxCollide;

    bvh_tree tree0 = bvh_tree(model0);
    bvh_tree tree1 = bvh_tree(model1);
    tree0.collide(&tree1, &BoxCollide);
    printf("collide box num : %d\n", BoxCollide.size());


    unsigned int id0, id1;
    vec3d p0;
    vec3d p1;
    vec3d p2;
    vec3d q0;
    vec3d q1;
    vec3d q2;

    for (int i = 0; i < BoxCollide.size(); i++) {
        BoxCollide[i].getTriPair(&id0, &id1);
        getTriangle(model0, id0, p0, p1, p2);
        getTriangle(model1, id1, q0, q1, q2);
        if (tri_contact(p0, p1, p2, q0, q1, q2)) {
            f->push_back(BVHIndexPair(id0,id1));
            model0->colors_[model0->faces_[id0].x] = vec3f(0.0f, 0.0f, 0.0f);
            model0->colors_[model0->faces_[id0].y] = vec3f(0.0f, 0.0f, 0.0f);
            model0->colors_[model0->faces_[id0].z] = vec3f(0.0f, 0.0f, 0.0f);
            model1->colors_[model1->faces_[id1].x] = vec3f(0.0f, 0.0f, 0.0f);
            model1->colors_[model1->faces_[id1].y] = vec3f(0.0f, 0.0f, 0.0f);
            model1->colors_[model1->faces_[id1].z] = vec3f(0.0f, 0.0f, 0.0f);
        }
    }
}
```

这是 execute 的代码，前面两个 for 循环用于三角形的着色，不需要关注。
首先进行 broad phase，也就是利用 bvh 先做一次剔除，将所有包围盒碰撞了的三角形 pair 先放到 BoxCollide 里面。
然后再 narrow phase，对 BoxCollide 里的每个三角形对进行非常精细的碰撞检测，结果存储到 CollidePair 里面。

这样就拿到了碰撞的三角面片，如果要进行碰撞响应，只需要对 CollidePair 进行操作就行