 DOI: https://doi.org/10.26549/gcjsygl.v4i6.3120

Research on Realization of Virtual Scene of Real Estate Based on OPENGL

Qingyang Zhang　Shuang Wang　Jinhui Huang

School of Computer and Information Engineering, Beijing Technology and Business University, Beijing, 100048, China

Abstract

In order to meet the current demand for the authenticity of the reproduction of virtual scenes while optimizing the viewer's visual experience, OpenGL is used to render the entire three-dimensional model to generate a true-color three-dimensional virtual scene. Starting from drawing simple 3D objects, gradually improving, and finally generating dynamic scenes, the information is completely and accurately converted into virtual scenes, so that visitors can experience the overall layout and various subtle items on the scene. At the same time, the combination of OpenGL and C++ increases the interactivity of the virtual scene and enhances the experience, allowing visitors to observe and manipulate the scene from different angles, and effective collision detection and processing make the virtual scene more realistic.

Keywords

OpenGL; glut; sky box model; collision detection and processing; bounding box algorithm

基于 OpenGL 对楼盘虚拟场景实现的研究

张清扬　王爽　黄今慧

北京工商大学计算机与信息工程学院，中国·北京 100048

摘　要

为满足目前对虚拟场景再现的真实性需求，同时优化观察者的视觉感受，利用 OpenGL 对整个三维模型进行渲染，生成真彩的三维虚拟场景。由绘制简单的 3D 物体开始，逐步完善，到最后生成动态场景的过程中，将信息完整、准确地转化为虚拟场景，使参观者能身临其境感受整体布局及各个细微项。同时 OpenGL 与 C++ 相结合，增加了虚拟场景的交互性，增强体验感，让参观者可以从不同角度来观察和操控场景，而有效的碰撞检测和处理让虚拟场景更加真实。

关键词

OpenGL；glut；天空盒模型；碰撞检测和处理；包围盒算法

1 引言

OpenGL，是一套底层图形库中的三维图形处理库，即开放性图形库 Open Graphics Library，针对三维图形可视化进行展开分析，是解决三维模型绘制、显示和交互问题的图形接口技术。它功能强大、调用方便，是一个跨编程语言、跨平台的编程接口。OpenGL 最初由 SGI 公司开发，作为图形工作站的一个强大的 3d 图形机制或图形标准，当初 SGI 公司为其图形工作站开发的 IRIS GL 是 OpenGL 的起源，随着跨平台移植，最终发展成为了 OpenGL。

OpenGL 应用广泛，在教学应用方面，它是高校交互式实验教学系统设计的基础。从最初的模型绘制，到最终的交互技术，都体现在了系统的各个模块中。模型认知模块主要应用了模型绘制和模型观察功能，建筑漫游模块则应用了光照、材质、纹理映射等功能，依托 OpenGL 完备的功能特性，才能构建起交互式试验系统。

而作为一种三维图形的开发标准，OpenGL 实现了高性能的三维图形开发。由 OpenGL 得到的图形质量好，性能高，稳定性好，着色方便；Open GL 可以完成对虚拟场景中对象的上色和渲染工作，让绘制的模型接近现实生活，具有更丰富的细节，得到尤为逼真的视觉效果。OpenGL 最基础的功能是模型绘制，通过基本绘图函数，进行二次封装，得到新的绘制函数，再由 OpenGL 的矩阵变换等操作，产生三维模型供实际应用，简单的如立方体、棱锥等；模型建立完成后，需要对模型进行观察，如设置视点、变换坐标、旋转模型等操作；接下来要为模型指定颜色，OpenGL 有 RGBA 模式和

 DOI: https://doi.org/10.26549/gcjsygl.v4i6.3120

颜色索引 (Color index) 两种物体着色方式。不同的颜色模式，其组合颜色的方式有所区别。基本变换和投影变换则是实现变换的两种主要方式。将 OpenGL 与 virtual studio 相结合，其变换的物体都是基于前一物体，在加上 OpenGL 提供的一系列图形转换函数，实现了基本变换。使用这种方法有利于提高三维图形的显示速度，与此同时在降低运行时间方面有一定的帮助。模型要产生阴影等效果，就用到了投影变换，此时，将借助光照的应用，OpenGL 的光照模型包括叠加各个独立的部分，即镜面光、环境光以及辐射光等光源，叠加之后，光照部分将有较快的速度，同时将看到更为明显的效果。而根据不同的光源，可以模拟多样化的光照环境；反走样和雾化可以解决图像的锯齿现象和远近层次关系，使之贴近真实；模型的细节层次，可以借助纹理映射技术来完善，通过链接真实图片，使三维景观更加逼真；从而动态流畅地显示三维模型。即虚拟场景中的模型绘制是实现场景的基础。

基于以上所述 Open GL 丰富全面的功能，我们希望能将其应用于楼盘虚拟场景的实现中[1]。目前房地产行业快速发展，而大多房地产商使用的平面效果图和沙盘形势单一，少数使用直升机参观又费时耗力成本大，为实现既能为参观者提供漫步于场景之中身临其境的真实感，又能全方位准确地观看楼房、小区设施及周围环境，让参观者经济有效快速地采集所需信息的效果，针对这些问题，本研究通过利用 Open GL 构筑虚拟场景，进行再现楼盘虚拟场景的探讨。

2 基于 OpenGL 的场景绘制

2.1 实验环境的搭建

实验开始前，先对实验环境进行配置。实验设备使用 visual studio 2017，首先新建一个 visual C++ 空项目，然后在项目中选择管理 Nuget 程序包，再浏览 glut 并选择安装 nupengl.core 包来完成配置。专为构建中小型 OpenGL 程序的应用工具包：glut，是一个和窗口系统无关的软件包，英文全称为 OpenGL Utility Toolkits，由 Mark Kilgard 任职于 SGI 公司时编写的。本次实验中，采用 glut 对虚拟场景进行绘制。

2.2 虚拟场景的绘制

首先，需要对楼市沙盘的信息进行采集。楼市的沙盘模型通常是按照一定的比例 (如 1000:1 比例尺) 对住宅小区中的建筑模型进行缩放，包括地形、绿植、河流湖泊、楼栋等信息。通过联系相应的房地产开发商，或对楼市沙盘进行直接拍照取样，获得相关的楼盘信息，即各个建筑模型的长宽高比例、模型与模型之间的相对位置关系等。对于本次实验，我们还希望能让用户可以进入楼栋内部，观察楼栋的内部大小和结构以及设计，因此将对楼盘室内信息进行采集。有了以上楼盘信息，我们就可以借助 OpenGL 来绘制楼盘的虚拟场景，接下来是针对具体实现虚拟场景部分的阐述。

OpenGL 提供了点、线、面的基本绘制函数，而在虚拟场景绘制中需要用到很多复杂的几何体的绘制函数，于是通过基本绘图函数，进行二次封装，得到新的绘制函数，再利用 OpenGL 提供的矩阵变换等一系列功能操作，产生具有一定几何外观的三维模型以供实际应用。现实中的建筑物大都可以看作长方体的堆叠，因而在进行场景构建前先对长方体进行封装，函数原型如下：

```
void drawCuboid(float x,  float y,  float z,
// 长方体顶点的 x,y,z 坐标
float a,  float b,  float h,
// 长方体的长、宽、高的值
float r,  float g,  float b,
// 长方体面颜色的 rgb 值
);
```

在函数体中调用填充四边形函数 glBegin(GL_QUADS);glEnd();，并使用 glVertex3f(); 绘点函数来顺序指定长方体的各个点坐标，同时可以用 glColor3f(); 函数对该物体进行着色。考虑到对文件容量和运行效率的控制，对于较为复杂的模型，需要减少绘制不必要的面，所以需要对模型构建和函数实现两方面进行优化。

在进行场景构建前，一方面，应考虑到虚拟场景中除主要绘制物体以外的环境，即场景中除模型以外的要素，如天空、场景外部、地形等。其中，地形模型对虚拟场景的真实性有重要的影响，而本次实验所选取的楼盘实景多为平坦的地形，较为简单，所以在实验中将地形平面化，用尺寸略大于场景的矩形平面代替地形，并进行纹理填充。另一方面，在实验中使用天空盒来增加虚拟场景的真实性，天空盒的实现思路是在场景中创建一个采用立方体贴图 (cubemap) 进行纹理采样的立方体，并始终让该立方体置于场景的外部，使观测者对天空产生触不可及的感觉。其中过程为：首先创建

 DOI: https://doi.org/10.26549/gcjsygl.v4i6.3120

cubemap，立方体贴图包含 6 个 2D 纹理，每一个 2D 纹理对应立方体的一个面。接着通过 SOIL 创建一个 cubemap，将其传入 shader，在着色器接收到纹理值后，再将纹理绘制出即可。

创建 cubemap：

```
GLuint mainTexture = SOIL_load_OGL_cubemap(
        "res/image/right.bmp",
        "res/image/left.bmp",
        "res/image/top.bmp",
        "res/image/bottom.bmp",
        "res/image/back.bmp",
        "res/image/front.bmp",
        0, 0, SOIL_FLAG_POWER_OF_TWO);
```

虚拟场景中的模型绘制是实现场景的基础，也是系统中相对复杂的部分，模型的选取将直接影响到虚拟场景的真实性。针对楼盘中的模型绘制，因为在楼盘中的物体主要是各种建筑设施，很少有形状复杂的物体，模型绘制相对简单，只需把楼盘小区中的物体抽象成基本几何体的组合，并按照一定的比例对尺寸进行缩放。最后再使用自定义的三维绘制函数在场景的指定位置进行绘制，就能完成对虚拟场景模型的绘制 [2]。

针对楼栋内部场景的绘制，要独立于之前所述的室外场景的绘制，将楼栋内部的场景位置设定在天空盒外部，让用户无法在室外场景中通过镜头移动到达内部场景。只有当用户与楼栋模型进行交互时，再调用渲染内部场景的函数，以达到减少程序渲染规模的效果。在调用内部渲染场景的函数时，会进行镜头参数更改，将镜头置于室内场景坐标内，同样当离开内部场景时释放资源。在渲染内部场景时，同样按照内部比例进行缩放，按照天空盒的思路，将镜头放在房间的内部，用平面的拼接的方法实现墙面绘制，再适当的绘制一下小物体，如家具和各种摆件。不同于普通的 OpenGL，glut 库中提供了许多特殊模型的绘制函数，我们可以用其来绘制复杂的小物体。

```
void glutSolidSphere(GLdouble radius, GLint slices, GLint stacks);
// 绘制实心球
void glutSolidTorus(GLdouble innerRadius, GLdouble outerRadius, GLint nsides, GLint rings);
// 绘制实心圆环
```

有了这些函数，室内场景中的物件更加丰富，从而给使用者提供更为全面的室内参考。

在系统中合理添加光照，也可以丰富虚拟场景。处理光照时，材质、光源和光照模型是 OpenGL 光照系统的三大部分。材质、光源和光照模式都有各自不同特质和属性，而这些可以通过函数来设置。可以利用 glLight 来设置的光源属性，包括光源的类型（环境光、漫反射光、镜面光）、光源的位置等。同时 glMaterial 还可以设置材质的属性。OpenGL 用材质对光源的 rgb 反射率来定义颜色 [3]。若光源的颜色为 (lr,lg,lb)，材质的颜色为 (mr,mg,mb)，则在忽略反射的情况下，真实呈现的颜色为 (lr*mr,lg*mg,lb*mb)。glLightModel 可以设置光照模型，其中全局环境光、镜面反射颜色、近视点或远视点、双面光照的设置，以及是否和环境颜色、散射颜色分开都属于光照模型的设置。在对光源进行操作时，可以用 glEnable 开启光源，glDisable 关闭光源。

3 虚拟场景的交互实现

交互是虚拟场景中重要的一环，虚拟场景的交互性即体现为用户对场景内物体的可操控程度和用户从场景中得到响应的自然程度，一个具有交互功能的虚拟场景可以让使用者可以从不同角度来观察和操控场景。

3.1 镜头和移动

OpenGL 为虚拟场景漫游提供了相关函数，在实现时主要使用 gluLookAt 函数对镜头进行操控，该函数的函数原型如下：

```
void gluLookAt(GLdouble eyex,GLdouble eyey,GLdouble eyez,
GLdouble centerx,GLdouble centery,GLdouble centerz,
GLdouble upx,GLdouble upy,GLdouble upz);
```

该函数定义了一个视图矩阵，并与当前矩阵相乘，其中的第一组参数 eyex,eyey,eyez 代表着镜头在世界坐标的位置，第二组参数 centerx,centery,centerz 代表着镜头对准的物体在世界坐标中的位置，第三组参数 upx,upy,upz 代表着镜头向上的方向在世界坐标中的位置。如果没有调用过 gluLookAt 函数，则默认情况下镜头的位置为世界坐标的原点，且指向 z 轴的负方向，朝上向量为 (0,1,0)。通过使用 gluLookAt 函数，我们可以模拟人眼在虚拟场景中的情况，借由函数参数的改变，实现了场景中的物体相对于观察者发生的变化，从而实现了

 DOI: https://doi.org/10.26549/gcjsygl.v4i6.3120

虚拟场景中的移动。

镜头的移动和转向应由使用者的操控而发生响应，在实验中采用键盘交互的方式来实现。通过使用键盘移动镜头，用向左向右键来控制镜头在 xoz 坐标平面围绕 y 轴进行旋转，向上向下键来控制镜头在当前朝向下的前进和后退，具体的实现方法如下 [4]。首先需要声明一些全局变量用来保存镜头的参数，其中包括镜头的位置和镜头指向目标方向的向量，同时还需要保存镜头的角度：

```
float angle;        // 镜头在 y 上旋转的角度
float x,z;          // 镜头在 xoz 坐标平面的位置
float lx,lz;        // 镜头的视线向量
```

接着处理箭头键，当使用者按下左右键时，角度变量 angle 也会随之改变。随着角度值的改变，程序将重新计算视线向量的 lx 和 lz 相应的值。由于镜头的位置只在 xoz 坐标平面上移动，所以不需要改变视觉向量 ly 的值，计算角度值和视线向量的公式如下：

```
lx = sin(angle)
lz = −cos(angle)
```

此时当更新 lx 和 lz 时，镜头的位置不会发生变化，变化的只有镜头指向物体的坐标。当移动镜头时，下一次的镜头位置需要沿着视线向量。为了达到这样的效果，需要在按下向上键或向下键时，加一或减一个粒度的视线向量到当前位置，例如在移动镜头向前时的计算公式如下：

```
x = x + lx * 粒度
z = z + lz * 粒度
```

此处的粒度可以理解为镜头移动的步幅，如果粒度值保持在一个常量，速率就会维持在一个常量范围内，增加粒度的值可以使速率更快，即每一帧移动的更远。

GLUT 库可以让程序自动监测到键盘输入，包括普通按键和特殊按键，并且提供两个函数来为键盘事件注册回调函数，实验中操控镜头时使用的箭头键属于特殊按键，所以应调用函数 glutSpecialFunc 进行注册。键盘事件的处理函数如下：

```
void pressKey(int key, int xx, int yy){
    switch (key){
    case GLUT_KEY_LEFT:
            deltaAngle = −0.04f;
            break;
    case GLUT_KEY_RIGHT:
            deltaAngle = 0.04f;
            break;
    case GLUT_KEY_UP:
            deltaMove = 0.2f;
            break;
    case GLUT_KEY_DOWN:
            deltaMove = −0.2f;
            break;
    default:
            break;
    }
}
```

其中的 deltaMove 表示移动的增量，deltaAngle 表示角度变化的增量。

当用户想要以俯视的角度观察场景时，可以通过按下键盘 1 键来让镜头处于场景上方并俯视场景。实现的方法是通过改变 gluLookAt 函数参数来实现，将 gluLookAt 函数中的 eyey 和 centery 两个参数用全局变量代替，当键盘 1 键按下事件发生时，通过改变全局变量的值，将镜头 y 坐标增加，使镜头置于场景上空，并将镜头指向物体的 y 坐标减小，从而达到俯视的效果。在俯视场景时把移动增量 deltaMove 的值设置为 0，阻止使用者在俯视时移动，防止在落回地面时镜头位置处于绘制模型的内部 [5]。

用户也可以通过与楼栋模型交互来进入室内场景，实现的方法是添加新的键盘监听事件，同时判定条件。当同时满足镜头坐标位于楼栋门口的矩形区域范围内时，调用渲染内部场景的函数，同时会将镜头坐标通过 gluLookAt 函数改变参数实现位移，用户在室内场景中同样可以通过上述的镜头移动算法来移动。

3.2 碰撞检测和处理

通过以上的方法实现了基本的虚拟场景漫游，但缺少对物理事件的模拟。当镜头位置与场景中模型所占据的虚拟空间重合时就会发生碰撞，为了增加虚拟场景的真实感，就要对碰撞进行检测并对检测结果做出处理。

虚拟场景对碰撞的解决通常需要使用包围盒算法。所谓包围盒算法，即是用体积稍大且特性简单的几何体对象来近

 DOI: https://doi.org/10.26549/gcjsygl.v4i6.3120

似地代替复杂的几何对象。同时，作为一种求解离散点集的最优包围空间算法，包围盒算法也是进行虚拟场景中碰撞干涉初步检测的重要方法。常见的包围盒算法有 AABB 包围盒（轴对齐包围盒）、包围球、方向包围盒 OBB 以及固定方向凸包 FDH，考虑到虚拟场景为小区楼宇，模型多为与坐标轴对齐的建筑物，所以采用 AABB 包围盒算法 [6]。在实验中，忽略观察者作为模型与场景中其他模型发生碰撞，将观察者看作一个点，只考虑其视点与其他模型的包围盒相碰撞的情况。碰撞的发生是由于视点进入了包围盒与实际模型之间的区域才产生的，当观察者前进时进入包围盒或后退时进入包围盒，即视线与包围盒面线段相交且视点坐标与包围盒重合时，判定为有碰撞发生。将包围盒封装为结构体，成员包括不同方向的范围值，由于实验中的移动只在 xoz 坐标平面上，所以成员只有上下左右四个方向的最大值和最小值。

碰撞检测函数如下：

```
bool isCollide(box *b){
    if ((b->leftMin >= x && b->leftMax <= x) ||
(b->rightMin >= x && b->rightMax <= x))
        return true;
    if ((b->upMin >= z && b->upMax <= z) ||
(b->downMin >= z && b->downMax <= z))
        return true;
    return false;
}
```

在程序中要对场景中的每一个模型的包围盒遍历检测函数，所有的物体无论与视点相距多远都要进行计算，这样会对性能产生较大的影响。更优解法是对模型进行排查，去掉当前不可能发生碰撞的模型，后续不进行相关的计算。在这里需要借助于一些用于场景管理的数据结构，例如八叉树、四叉树等，其思路是用递归来划分空间，最终确定模型所在的空间。以上是实验碰撞检测的改进方案。

在碰撞发生之后，就要做出相应的碰撞响应。当碰撞发生时，将视点恢复到碰撞发生前一瞬间的位置，在视点与边缘相交时恢复到恰好不相交的位置，而在过程中保持角度不发生改变。由于主函数中的渲染函数 glutDisplayFunc() 为循环调用，所以需要在每一帧镜头移动后进行碰撞检测，并对发生碰撞的分量取上一步的值。碰撞处理的流程图如下。

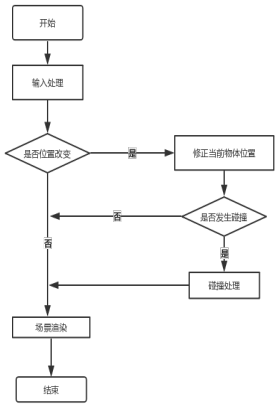

图 1 碰撞流程处理图

有了碰撞检测后，虚拟场景中的交互更加真实，用户的沉浸感也随之增加。

4 结语

OpenGL 凭借其强大的 2D、3D 图形渲染能力，在图形仿真、虚拟实现等领域上起着重要作用。本次实验利用 OpenGL 完成了对楼盘虚拟场景的搭建，并结合 C++ 代码增加了场景的交互性。在实验中克服了室内搭建、光照、碰撞等难点，最终实现了楼盘的虚拟场景，使用户可以对楼盘的不同角度进行观察，从室内到楼栋外部，形成了一个全面的观察范围。在虚拟楼盘中漫游可做为用户购房的参考，实现了实验的预期目标。

参考文献

[1] 张亚莉 . 基于 OpenGL 的虚拟校园漫游系统 [D]. 西安 : 西安工业大学 .2017:8-19.

[2] ZJU_fish1996.[OpenGL] 基于 AABB 包围盒的漫游时视点与场景的碰撞检测 [EB/OL].(2016-7-10)[2019-7-18].https://blog.csdn.net/ZJU_fish1996/article/details/51869828\.

[3] qq_38415161. 包围盒算法 [EB/OL].(2017-11-10)[2019-7-18]. https://blog.csdn.net/qq_38415161/article/details/78499967.

[4] 进击的巨人王 .OpenGL 中的光照、材质等属性 [EB/OL].(2015-10-28)[2019-7-17].https://blog.csdn.net/wang15061955806/article/details/49469019.

[5] 如云缥缈 . 初学 OpenGL(7): 颜色和光照 [EB/OL].(2018-5-3)[2019-7-16].https://blog.csdn.net/qq_35370018/article/details/80188116.

[6] The fool. OpenGL 学习脚印 : 立方体纹理和天空包围盒 (Cubemaps And Skybox)[EB/OL].(2016-9-11)[2019-7-16]. https://blog.csdn.net/wangdingqiaoit/article/details/52506893.