# 2月27日 星期四

今日完成任务：学习使用Qt开发图形界面软件与PBR部分理论的学习

今天学习了Qt一些组件的使用和Qt开发中信号与槽的概念。Qt中信号与槽是关键机制，是一种高级接口，应用于对象之间的通信。一般的GUI工具都是用回调函数来解决图形界面的响应，但这种解决方式会导致许多凌乱的函数指针的出现，使得程序不够简洁明了。Qt中使用的信号与槽机制取代了函数指针，使得界面响应的实现更加清晰明了。信号与槽机制将信息封装起来，槽并不知道是否有任何信号与自己相连接。

PBR部分学习了《全局光照》一书上关于渲染方程的部分，本日主要学习的是物理渲染的三个条件，微平面、能量守恒和BRDF。其中，重点学习了BRDF部分。一个表面对不同的光线入射角和反射角的组合，拥有不同的反射率，而BRDF正是通过输入的入射角与反射角来反映出当前表面的反射性质。BRDF可以在片元着色器中进行逐像素计算，不必一定使用光线追踪的相关算法。

# 2月28日 星期五

今日完成：Qt图形界面框架的学习与PBR理论学习

今日学习了Qt提供的界面搭建功能，并尝试搭建渲染器界面。Qt Designer是个十分方便的GUI构造器，它会生成.ui结尾的设计文件，然后该文件最终会编译成.h文件。Qt的这部分功能相比同样是图形界面的框架的MFC，Qt Designer对于设计者十分友善。MFC几乎没有图形界面的界面设计功能，几乎所有的设计都得在代码中进行，预览只能靠编译之后的运行才能看见。而Qt Designer则有预览界面，甚至能查阅程序自动生成的代码，相比MFC就要方便直观地多。

今日PBR相关学习了Cook-Torrance反射率方程。该方程为一个具体化的渲染方程，该方程具体化了漫反射部分与镜面反射部分。在镜面反射中菲涅尔方程部分为了节省计算开销做了近似处理。Cook-Torrance反射率方程完整的描述了一个基于物理的渲染模型，根据这个模型可以拟定PBR材质需要控制哪些属性。

# 2月29日 星期六

今日完成：Qt图形框架中需要用到的组件尝试性使用与学习

Qt提供了许多使用的界面元素的封装，如按钮、菜单栏、工具栏、窗口等。每种功能性的界面元素都对应Qt中的一个类，这些类有着不同的信号与槽函数。今日尝试编写了些基础代码实验并熟悉可能会使用的信号与槽函数。在测试过程中法线，信号与槽的机制比想象中更为简单明了。信号的响应对象可以不限定于槽上，信号还可以触发另一个信号，这可以让程序的界面调用结构更加清晰。并且信号与槽机制与设计模式中观察者模式十分近似，所以这个机制有着观察者模式的优点。然后还尝试着实现了些自定义的信号和槽函数。槽函数与类中普通的公有函数的功能差不多，甚至直接使用公有函数当作槽函数也行。估计是之前的Qt版本是不行，但槽函数也是提供一个函数的地址，所以没差。但还是将公有函数和槽函数区分才行，这样能保证程序结构的清晰。

# 3月2日 星期一

今日完成：文献翻译

在翻译的基本理论里面，有很多很多能用的翻译技巧。直译，它的定义目前尚无定论，但是可以采用一种说法，即在不违背译文语言规范以及不引起错误联想的前提下，在译文中既保留原文内容又保留原文形式，特别指保持原文的比喻、形象和民族地方色彩等的译文为直译。目标是作到形式与内容的统一，即“形”，“神”兼备。意译，简单地说，是放弃原文的修辞格，也不用其他修辞格，只是把意思平铺直叙地翻译出来。他是通过对原词汇深层蕴意的理解将原词汇的表层结构有机地转化为译语的表层结构，重在再现原词汇的蕴意，真实的传达词汇所含的信息。

# 3月3日星期二

今日完成：文献翻译

科技文章的遣词也可以生动活泼,细究之下,似有三类:借、转、破。它们给可能单调的行文平添几分亮色,使科技文章的沉滞之风,为之一扫,于是阅读便成为享受,翻译也充满兴趣。 科技文章造句也讲究文采，细细品味科技文章，主要体现了三中美、人性美、练达美、逻辑美。 科技英语最具特色的美就是逻辑美，因为科技英语是表达科技事实、概念、原理的，和解释自然现象的，因此逻辑缜密，推导合理，无懈可击。英语翻译是目前世界翻译量最大的语种，在国内英语占据了85%的翻译量，各类资料的翻译都是涉及英语与中文之间的转换。

# 3月4日星期三

今日完成：文献翻译

论文翻译往往会涉及到很生僻的词语和合成词，有些词语很有可能在传统的书籍词典，电子词典中查不到。翻译用语要符合论文习惯，必要时请擅长论文翻译的专业人士校对或者是翻译，比如高校翻译老师等。不同的场合背景下，句子表达有着不同的方式。在论文的语境下，翻译更多地要注重言简意赅、用词到位不花哨，校对也是不可缺少的环节。论文翻译时一定要强调专业性。闻到有先后，术业有专攻。即使翻译的水平再高，但是如果不理解论文所在的领域，也很难将论文翻译出色。

# 3月5日 星期四

今日完成：学习了Qt图形界面框架的OpenGL部分并做点实验

在Qt中有对OpenGL的封装。QOpemGLWidget类为OpenGL在多个操作系统平台上的应用进行了浅层封装。此外，Qt还提供加载图像并将其与纹理绑定的使用功能，还有一套自己的3D数学库。之前使用的是OpenGL原生库和glm库进行的OpenGL相关实现，所以今日对Qt中的OpenGL进行了一些实验性的编写。Qt自带的3D数学库相比较于glm库，在一些运算方面不是很直观。glm库中的类定义于C++自带的数据类型很相似，使用起来是很直觉性的。而Qt中的3D数学库感觉封装过于严密，有点偏离C++的感觉。但Qt中对OpenGL的封装是成一个体系的，许多函数脱离不了自带的数学库。不过，学习中发现Qt中是可以使用原生OpenGL代码的，但不知道对性能方面是否有所影响。尝试用QOpemGLWidget做了个简单物体的渲染。

# 3月6日 星期五

今日完成：PBR理论学习和Qt图形界面的组件使用

今日主要看了看IBL相关知识点。IBL运用于光照模型中的间接光照部分，主要思路就是预计算，把复杂的积分都先算好。IBL中会分别预计算漫反射项和镜面项，最终在实时渲染中只需通过简单的纹理采样即可得到结果。Cook-Torrance模型中分为了漫反射和镜面反射两个部分。漫反射部分的积分项为一个常数项，将其移出积分后可以对计算点所在的半球空间进行离散采样取平均值来计算出积分结果。这个计算过程是可以进行预计算，以提高运行时的效率。

今日依旧时学习使用Qt的QOpenGL Widget。这个部分跟OpenGL原生代码的区别不算太大。Qt将整个窗口组件的生命周期分为初始化与渲染循环两个部分。在渲染循环相当于设置每一帧需要渲染什么，初始化主要完成对OpenGL的配置。Qt的3D数学库依旧，很难用。

# 3月7日 星期六

今日完成：文献翻译

术语一致性在技术翻译中尤为重要，同时技术文件存在一定重复性的特点，并且日后可能还需不时进行文字更新，计算机辅助翻译工具很好地满足了这些需要，它具备功能强大的翻译记忆库和术语管理数据库，可以确保术语的一致性和翻译的高效率。技术翻译的任务在于准确而完整地介绍原作的思想内容,使读者对原作的思想内容有正确的理解。对于科学技术文献和图纸的翻译,尤其需要如此。因为科技作品的任务在于准确而系统地论述科学技术问题,它要求高度的准确性。因此,对科技翻译要特别强调准确性,尤其是对术语、定义、定理和结论的翻译应予以特别的重视,另外,科技文体要注重客观性,不做主观渲染,尽量不用带感情色彩的词汇,不需含蓄与夸张,做到说理叙事明白清楚,用词造句简洁明确。

指导教师评语

# 3月9日 星期一

今日完成：IBL漫反射部分的学习与理解

在 PBR 渲染管线中考虑HDR的场景光照非常重要。由于 PBR 的大部分输入基于实际物理属性和测量，因此为入射光值找到其物理等效值是很重要的。无论我们是对光线的辐射通量进行研究性猜测，还是使用它们的直接物理等效值，诸如一个简单灯泡和太阳之间的这种差异都是很重要的，如果不在 HDR 渲染环境中工作，就无法正确指定每个光的相对强度。在毕设中就要考虑读取与解析hrd文件。hrd文件存储了一张完整的立方体贴图，所有六个面数据都是浮点数。它并非直接存储每个通道的 32 位数据，而是每个通道存储 8 位，再以 alpha 通道存放指数。stb\_image库可以读取普通的图片作为贴图的同时，还可以读取hdr文件。由于hdr文件中存入的是等距柱状投影图，还需要将其转换为立方贴图，这样方便操作一些。毕设过程中也可探究直接在等距柱状投影图上采样快还是在立方贴图上采样快。

# 3月10日 星期二

今日完成：学习IBL的镜面反射部分

Cook-Torrance 镜面部分在整个积分上不是常数，不仅受入射光方向影响，还受视角影响。如果试图解算所有入射光方向加所有可能的视角方向的积分，二者组合数会极其庞大，实时计算太昂贵。这次毕设中应该采用epic提出的分割求和近似法，将方程的镜面部分分割成两个独立的部分，我们可以单独求卷积，然后在 PBR 着色器中求和，以用于间接镜面反射部分 IBL。卷积的第一部分被称为预滤波环境贴图，是预先计算的环境卷积贴图，但其中考虑了粗糙度。随着粗糙度的增加，参与环境贴图卷积的采样向量会更分散，导致反射更模糊。第二部分等于镜面反射积分的 BRDF 部分。如果我们假设每个方向的入射辐射度都是白色的（因此 ），就可以在给定粗糙度、光线 法线 夹角 的情况下，预计算 BRDF 的响应结果。将预计算好的 BRDF 对每个粗糙度和入射角的组合的响应结果存储在一张 2D 查找纹理(LUT)上，2D 查找纹理存储是菲涅耳响应的系数（R 通道）和偏差值（G 通道）。

# 3月11日 星期三

今日完成：IBL镜面反射部分的学习

今天主要学习了下蒙特卡洛积分相关。在渲染方程中的被积函数往往比较复杂，不能简单求出数值解。蒙特卡洛积分的目的是对积分进行近似计算。蒙特卡洛方法归纳为以下步骤：

对满足某一概率分布的随机数进行抽样；使用抽样值计样本贡献值；。对所有的抽样结果求平均值得到积分的估计。蒙特卡罗积分建立在大数定律的基础上，并采用相同的方法来求解积分。不为所有可能的样本值 求解积分，而是简单地从总体中随机挑选样本 生成采样值并求平均。随着样本数的增加，结果会越来越接近积分的精确结果。某些蒙特卡洛估算是有偏的，这意味着生成的样本并不是完全随机的，而是集中于特定的值或方向。这些有偏的蒙特卡洛估算具有更快的收敛速度，它们会以更快的速度收敛到精确解，但是由于其有偏性，可能永远不会收敛到精确解。但在计算机图形学中。因为只要结果在视觉上可以接受，解决方案的精确性就不太重要。

# 3月12日 星期四

今日完成：需求分析与文档写作

一款展示从传统Blinn-Phong光照模型PBR的变化过程的渲染器。将PBR材质、直接光照、HDR、间接漫反射和IBL间接镜面反射做为递进式开放的功能，当完成当前部分的学习后就可解锁对应的功能，最终解锁为一个完整的PBR渲染器，并且使用平台定位windows。图形接口为OpenGL，界面框架使用Qt来进行开发。软件可以导入模型，默认提供立方体模型、球形模型与圆柱模型。可以对选中的物体移动其位置、旋转与调整其大小。今日需求分析文档并没有完成，明日会继续完成。

# 3月13日 星期五

今日完成：需求分析文档，系统设计

完成了昨日剩下的需求文档的编写。不画 UML 类图的话，复杂一点的框架都是无法设计的，在网上寻找了些UML绘制工具。最终选择使用Visual Paradigm来绘制UML图。Visual Paradigm Community Edition是自2004年以来推出的，旨在提供免费的UML软件，用于非商业目的，支持在UML建模方面迈出第一步的用户，以及需要免费的跨平台UML建模软件的用户个人使用，如在学生项目中使用UML。 UML建模工具免费用于各种非商业目的。今日主要在试用各种UML工具，系统设计并没有推进太多。

# 3月16日 星期一

今日完成：系统设计和设计文档写作

进行系统设计时，必须把所要设计的对象系统和围绕该对象系统的环境共同考虑，前者称为内部系统，后者称为外部系统，它们之间存在着相互支持和相互制约的关系，内部系统和外部系统结合起来称作总体系统。因此，在系统设计时必须采用内部设计与外部设计相结合的思考原则，从总体系统的功能、输入、输出、环境、程序、人的因素、物的媒介各方面综合考虑，设计出整体最优的系统。进行系统设计应当采用分解、综合与反馈的工作方法。不论多大的复杂系统，首先要分解为若干子系统或要素，分解可从结构要素、功能要求、时间序列、空间配置等方面进行，并将其特征和性能标准化，综合成最优子系统，然后将最优子系统进行总体设计，从而得到最优系统。在这一过程中，从设计计划开始到设计出满意系统为止，都要进行分阶段及总体综合评价，并以此对各项工作进行修改和完善。整个设计阶段是一个综合性反馈过程。

# 3月17日 星期二

今日完成：系统设计和设计文档写作

系统设计通常应用两种方法：一种是归纳法，另一种是演绎法。应用归纳法进行系统设计的程序是：首先尽可能地收集现有的和过去的同类系统的系统设计资料；在对这些系统的设计、制造和运行状况进行分析研究的基础上，根据所设计的系统的功能要求进行多次选择，然后对少数几个同类系统作出相应修正，最后得出一个理想的系统。演绎法是一种公理化方法，即先从普遍的规则和原理出发，根据设计人员的知识和经验，从具有一定功能的元素集合中选择能符合系统功能要求的多种元素，然后将这些元素按照一定形式进行组合（见系统结构），从而创造出具有所需功能的新系统。在系统设计的实践中，这两种方法往往是并用的。

# 3月18日 星期三

今日完成：软件界面原型设计

Mockplus是一款高效简单的app界面设计工具，产品经理圈子中比较流行的一款国产界面原型设计工具。软件提供了封装好的3000多个图标以及200多个组件，拖拽即可进行UI界面设计，学习成本低，无需代码参与，所以对于UI设计师来说是一个非常好的选择。Mockplus项目树层级很分明，项目树中以分组和子页面来规定层级。严格的层级管理有利于梳理逻辑，让原型项目层次清晰。同时也展现了Mockplus在交互上的功能性。另外，Mockplus还支持导出项目树，可导出为：树图、脑图、html、Csv（用Excel直接打开）、XML、文本、MarkDown文本等，使项目文档制作更加方便。

# 3月19日星期四

今日完成：类设计

本次毕设软件将采取MVC设计模式。MVC开始是存在于桌面程序中的，M是指业务模型，V是指用户界面，C则是控制器，使用MVC的目的是将M和V的实现代码分离，从而使同一个程序可以使用不同的表现形式。比如一批统计数据可以分别用柱状图、饼图来表示。C存在的目的则是确保M和V的同步，一旦M改变，V应该同步更新。MVC指MVC模式的某种框架，它强制性的使应用程序的输入、处理和输出分开。使用MVC应用程序被分成三个核心部件：模型、视图、控制器。它们各自处理自己的任务。框架、设计模式这两个概念总容易被混淆，其实它们之间还是有区别的。框架通常是代码重用，而设计模式是设计重用，架构则介于两者之间，部分代码重用，部分设计重用，有时分析也可重用。在软件生产中有三种级别的重用：内部重用，即在同一应用中能公共使用的抽象块;代码重用，即将通用模块组合成库或工具集，以便在多个应用和领域都能使用；应用框架的重用，即为专用领域提供通用的或现成的基础结构，以获得最高级别的重用性。

指导教师评语

# 3月20日 星期五

今日完成：类图绘制

类图是显示了模型的静态结构，特别是模型中存在的类、类的内部结构以及它们与其他类的关系等。类图不显示暂时性的信息。类图是面向对象建模的主要组成部分。它既用于应用程序的系统分类的一般概念建模，也用于详细建模，将模型转换成编程代码。类图也可用于数据建模。类图主要用在面向对象软件开发的分析和设计阶段，描述系统的静态结构。类图图示了所构建系统的所有实体、实体的内部结构以及实体之间的关系。即．类图中包含从用户的客观世界模型中抽象出来的类、类的内部结构和类与类之间的关系。它是构建其他设计模型的基础，没有类图，就没有对象图、状态图、协作图等其他UMI．动态模型图．也就无法表示系统的动态行为。类图也是面向对象编程的起点和依据。

# 3月23日 星期一

今日完成：用例图绘制

用例图是指由参与者（Actor）、用例（Use Case），边界以及它们之间的关系构成的用于描述系统功能的视图。用例图（User Case）是外部用户（被称为参与者）所能观察到的系统功能的模型图。用例图是系统的蓝图。用例图呈现了一些参与者，一些用例，以及它们之间的关系，主要用于对系统、子系统或类的功能行为进行建模。用例是对包括变量在内的一组动作序列的描述，系统执行这些动作，并产生传递特定参与者的价值的可观察结果。简单来说，用例是参与者想要系统做的事情。用例图主要的作用有三个：（1）获取需求；（2）指导测试；（3）还可在整个过程中的其它工作流起到指导作用。用例图描述的是参与者所理解的系统功能，主要元素是用例和参与者，是帮助开发团队以一种可视化的方式理解系统的功能需求。这时处于项目初始，分析用户需求的阶段，不用管怎么实现具体的功能，只要能向客户形象化的表述项目的功能就行。

# 3月24日 星期二

活动图是阐明了业务用例实现的工作流程。业务工作流程说明了业务为向所服务的业务主角提供其所需的价值而必须完成的工作。业务用例由一系列活动组成，它们共同为业务主角生成某些工件。工作流程通常包括一个基本工作流程和一个或多个备选工作流程。工作流程的结构使用活动图来进行说明。业务用例工作流程说明了业务为向所服务的业务主角提供其所需的价值而必须完成的工作。业务用例由一系列活动组成，它们共同为业务主角生成某些工件。工作流程通常包括一个基本工作流程和一个或多个备选工作流程。工作流程的结构使用活动图来进行说明。工作流程活动图用于研究实现业务目标时所要执行的各项任务或活动的顺序安排。活动既可以是手动执行的任务，也可以是自动执行的任务。它可完成一个工作单元。活动图是状态图的一种特殊形式。其中所有或多数状态都是活动状态，而且所有或多数转移都在源状态中的活动完成时立即触发。

# 3月25日 星期三

今日完成：状态图

状态图(Statechart Diagram)是描述一个实体基于事件反应的动态行为，显示了该实体如何根据当前所处的状态对不同的事件做出反应。通常我们创建一个UML状态图是为了以下的研究目的：研究类、角色、子系统、或组件的复杂行为。状态机用于对模型元素的动态行为进行建模，更具体地说，就是对系统行为中受事件驱动的方面进行建模（请参见概念：事件与信号）。状态机专门用于定义依赖于状态的行为（即根据模型元素所处的状态而有所变化的行为）。其行为不会随着其元素状态发生变化的模型元素不需要用状态机来描述其行为（这些元素通常是主要负载管理数据的被动类）。状态机由状态组成，各状态由转移链接在一起。状态是对象执行某项活动或等待某个事件时的条件。转移是两个状态之间的关系，它由某个事件触发，然后执行特定的操作或评估并导致特定的结束状态。

# 3月26日 星期四

今日完成：完成详细设计文档

详细设计，是软件工程中软件开发的一个步骤，就是对概要设计的一个细化，就是详细设计每个模块实现算法，所需的局部结构。在详细设计阶段，主要是通过需求分析的结果，设计出满足用户需求的软件系统产品。传统软件开发方法的详细设计主要是用结构化程序设计法。详细设计的表示工具有图形工具和语言工具。图形工具有业务流图、程序流程图、PAD图（Problem Analysis Diagram）、NS流程图（由 Nassi和 Shneidermen开发，简称 NS）。语言工具有伪码和PDL（Program Design Language）等。基本任务(1)为每个模块进行详细的算法设计。用某种图形、表格、语言等工具将每个模块处理过程的详细算法描述出来。(2)为模块内的数据结构进行设计。对于需求分析、概要设计确定的概念性的数据类型进行确切的定义。(3)为数据结构进行物理设计，即确定数据库的物理结构。物理结构主要指数据库的存储记录格式、存储记录安排和存储方法，这些都依赖于具体所使用的数据库系统。

# 3月27日 星期五

今日完成：渲染器基础框架搭建

将整个程序的几个大类构建了起来。Render类负责管理OpenGL接口来进行场景渲染。Render类继承Qt提供的QOpenGLWidget类，QOpenGLWidget类是用于渲染OpenGL图形。QOpenGLWidget类提供了在Qt应用程序中显示OpenGL图形的功能。它使用起来非常简单：新建类继承于QOpenGLWidget，使用方法就像继承于QWidget类子类一样。paintGL()—渲染OpenGL场景，需要更新Widget时就会调用。resizeGL()—设置OpenGL视口，投影等。每当调整Widget的大小时（第一次显示窗口Widget时会调用它）。

initializeGL()—建立OpenGL的资源和状态。在第一次调用resizeGL()或paintGL()之前调用一次。如果需要从paintGL()以外的地方触发重绘（一个典型的例子是使用定时器为场景设置动画），应该调用widget的update()函数来进行更新。当调用paintGL()，resizeGL()或initializeGL()时，Widget的OpenGL渲染环境需要设为当前。如果需要从其他位置调用标准OpenGL API函数（例如，Widget的构造函数或自己的绘图函数中），则必须首先调用makeCurrent()。所有渲染都发生在OpenGL帧缓冲对象中，makeCurrent()确保它在渲染环境中，在paintGL()中的渲染代码中创建和绑定其他帧缓冲对象时，不要使用ID 0重新绑定帧缓冲区，而是调用defaultFramebufferObject()来获取应该绑定的ID。

# 3月30日 星期一

今日完成：渲染器框架搭建

PBR\_Renderer类为整个渲染器的界面类。这个类继承于QMainWindow。QMainWindow类提供一个有菜单条、工具栏、状态条的主应用程序窗口（例如：开发Qt常用的IDE-Visual Studio、Qt Creator等）。一个主窗口提供了构建应用程序的用户界面框架。Qt拥有QMainWindow及其相关类来管理主窗口。QMainWindow拥有自己的布局，我们可以使用QMenuBar（菜单栏）、QToolBar（工具栏）、QStatusBar（状态栏）以及QDockWidget（悬浮窗体），布局有一个可由任何种类小窗口所占据的中心区域。

# 3月31日 星期二

今日完成：渲染器框架搭建

今日完成Camera类和RenderObject类。Camera类负责场景渲染中相机数据的保存和坐标转化矩阵的提供。同时Camera还得能在场景中移动。当讨论摄像机/观察空间(Camera/View Space)的时候，是在讨论以摄像机的视角作为场景原点时场景中所有的顶点坐标：观察矩阵把所有的世界坐标变换为相对于摄像机位置与方向的观察坐标。要定义一个摄像机，我们需要它在世界空间中的位置、观察的方向、一个指向它右测的向量以及一个指向它上方的向量。细心的读者可能已经注意到我们实际上创建了一个三个单位轴相互垂直的、以摄像机的位置为原点的坐标系。使用矩阵的好处之一是如果你使用3个相互垂直（或非线性）的轴定义了一个坐标空间，可以用这3个轴外加一个平移向量来创建一个矩阵，并且可以用这个矩阵乘以任何向量来将其变换到那个坐标空间。

# 4月1日 星期三

今日完成：渲染器框架搭建

RenderObject是整个渲染器所有物体的父类，方便Render类管理与使用。今日同时完成了立方体物体与球体的绘制与管理。立方体可以通过直接在代码中写死数据的方式进行绘制。球体的绘制则要复杂一些。球面的参数曲线可以用球坐标表示，引入参数u,v，其中v是球面点与原点的连线与z轴正向的夹角，u表示连线在xy平面的投影与x轴正向的夹角。已知球面的参数方程以后，很容易求得给定点的法向量，分别对u和v方向求偏导数，然后对两个所得向量进行叉积即可。已知参数方程以后，需要进行离散，分别设定u和v的步长：ustep、vstep。然后通过不同的u和v，求得坐标系中点的实际坐标(x,y,z)

指导教师评语

# 4月2日 星期四

今日完成：渲染器搭建

今日完成几个基础shader的编写，传统光照模型有贴图版本和无贴图版本。今日默认场景中自由平行光，在shader中只计算有关平行光的属性。现实世界的光照是极其复杂的，而且会受到诸多因素的影响，这是我们有限的计算能力所无法模拟的。因此OpenGL的光照使用的是简化的模型，对现实的情况进行近似，这样处理起来会更容易一些，而且看起来也差不多一样。这些光照模型都是基于我们对光的物理特性的理解。其中一个模型被称为冯氏光照模型(Phong Lighting Model)。冯氏光照模型的主要结构由3个分量组成：环境(Ambient)、漫反射(Diffuse)和镜面(Specular)光照。光通常都不是来自于同一个光源，而是来自于我们周围分散的很多光源，即使它们可能并不是那么显而易见。光的一个属性是，它可以向很多方向发散并反弹，从而能够到达不是非常直接临近的点。所以，光能够在其它的表面上反射，对一个物体产生间接的影响。考虑到这种情况的算法叫做全局照明(Global Illumination)算法，但是这种算法既开销高昂又极其复杂。

# 4月3日 星期五

今日完成：渲染器搭建

添加对其他类型的光源的支持，同时添加多光源的支持。为了在场景中使用多个光源，希望将光照计算封装到GLSL函数中。这样做的原因是，每一种光源都需要一种不同的计算方法，而一旦想对多个光源进行光照计算时，代码很快就会变得非常复杂。如果只在main函数中进行所有的这些计算，代码很快就会变得难以理解。GLSL中的函数和C函数很相似，它有一个函数名、一个返回值类型，如果函数不是在main函数之前声明的，我们还必须在代码文件顶部声明一个原型。我们对每个光照类型都创建一个不同的函数：定向光、点光源和聚光。每个光源类型都将它们的贡献加到了最终的输出颜色上，直到所有的光源都处理完了。最终的颜色包含了场景中所有光源的颜色影响所合并的结果。

# 4月6日 星期一

今日完成：渲染器搭建

今日实现对模型文件的读取功能。OBJ文件是一种标准3D模型文件格式，适合用于3D软件模型之间的互导。OBJ文件是一种文本文件，可以直接用写字板打开进行查看、编辑和修改，由一行行文本组成，注释行以符号“#”为开头，数据行由一两个标记字母也就是关键字开头，关键字解释了数据类型。Assimp能够导入很多种不同的模型文件格式，它会将所有的模型数据加载至Assimp的通用数据结构中。当Assimp加载完模型之后，就能够从Assimp的数据结构中提取所需的所有数据了。由于Assimp的数据结构保持不变，不论导入的是什么种类的文件格式，它都能够从这些不同的文件格式中抽象出来，用同一种方式访问需要的数据。当使用Assimp导入一个模型的时候，它通常会将整个模型加载进一个场景(Scene)对象，它会包含导入的模型/场景中的所有数据。

# 4月7日 星期二

今日完成：渲染器搭建

今日继续完成模型读取的实现。当用户通过操作用户界面发送了添加模型的信号后，Render类中的AddModel槽函数进行响应。同时，该信号中应该传送模型的地址。由于渲染是实时进行的，如果实时将模型数据添加到场景物体数组中，将不能确定这个过程究竟发生在渲染的哪一步。这很有可能使得在渲染过程中读取到脏数据。所以，AddModel将会设置Render的一个添加模型的标志布尔变量，然后在下一帧渲染开始时将模型数据添加到场景物体数据数组中进行渲染。

对于通过Assimp读取的模型文件，将通过Model类和Mesh类进行处理。Model类中将存储模型的所有Mesh和贴图数组。Mesh中将会存储读取的顶点数据和索引数据来用于渲染。同时，在处理模型文件的过程中将会为每个Mesh创建VAO、VBO和EBO。然后，在渲染时，Render类将会统一调用Model的父类定义的Draw函数。由于是虚函数，实际上调用Model实现的Draw函数，在Model的Draw函数中将会负责将每个Mesh进行绘制。

# 4月8日 星期三

今日完成：渲染器搭建

今日实现PBR材质。为了满足反射率方程，我们循环遍历每一个光源，计算他们独立的辐射率然后求和，接着根据BRDF和光源的入射角来缩放该辐射率。我们可以把循环当作在对物体的半球领域对所以直接光源求积分。由于是线性空间内计算光照，使用在物理上更为准确的平方倒数作为衰减。然后，对于每一个光源都计算完整的 Cook-Torrance specular BRDF项：首先计算的是镜面反射和漫反射的系数，或者说发生表面反射和折射的光线的比值；然后计算剩下的正态分布函数D和几何遮蔽函数G。最终的结果Lo，或者说是出射光线的辐射率，实际上是反射率方程的在半球领域Ω的积分的结果。以上所有计算都在线性的颜色空间中进行的，因此需要在着色器最后做伽马矫正。在线性空间中计算光照是非常重要的，因为PBR要求所有输入都是线性的，如果不是这样，就会得到不正常的光照。另外，希望所有光照的输入都尽可能的接近他们在物理上的取值，这样他们的反射率或者说颜色值就会在色谱上有比较大的变化空间。

# 4月9日 星期四

今日完成：渲染器搭建

今日实现带贴图的PBR材质。为了能够把纹理映射(Map)到三角形上，需要指定三角形的每个顶点各自对应纹理的哪个部分。这样每个顶点就会关联着一个纹理坐标(Texture Coordinate)，用来标明该从纹理图像的哪个部分采样。之后在图形的其它片段上进行片段插值(Fragment Interpolation)。stb\_image.h是Sean Barrett的一个非常流行的单头文件图像加载库，它能够加载大部分流行的文件格式，并且能够很简单得整合到工程之中。glGenTextures函数首先需要输入生成纹理的数量，然后把它们储存在第二个参数的unsigned int数组中（我们的例子中只是单独的一个unsigned int），就像其他对象一样，我们需要绑定它，让之后任何的纹理指令都可以配置当前绑定的纹理。纹理已经绑定后，使用载入的图片数据生成一个纹理，纹理可以通过glTexImage2D来生成。当调用glTexImage2D时，当前绑定的纹理对象就会被附加上纹理图像。目前只有基本级别(Base-level)的纹理图像被加载了，如果要使用多级渐远纹理，必须手动设置所有不同的图像。

# 4月10日 星期五

今日完成：渲染器搭建

今日实现HDR文件的读入与处理。辐射度文件的格式（扩展名为 .hdr）存储了一张完整的立方体贴图，所有六个面数据都是浮点数，允许指定 0.0 到 1.0 范围之外的颜色值，以使光线具有正确的颜色强度。这个文件格式使用了一个聪明的技巧来存储每个浮点值：它并非直接存储每个通道的 32 位数据，而是每个通道存储 8 位，再以 alpha 通道存放指数——虽然确实会导致精度损失，但是非常有效率，不过需要解析程序将每种颜色重新转换为它们的浮点数等效值。stb\_image.h 支持将辐射度 HDR 图像直接加载为一个浮点数数组。stb\_image.h 自动将 HDR 值映射到一个浮点数列表：默认情况下，每个通道32位，每个颜色 3 个通道。

# 4月13日 星期一

今日完成：渲染器搭建

今日实现HDR文件的读入与处理。要将等距柱状投影图转换为立方体贴图，需要渲染一个单位立方体，并从内部将等距柱状图投影到立方体的每个面，并将立方体的六个面的图像构造成立方体贴图。此立方体的顶点着色器只是按原样渲染立方体，并将其局部坐标作为 3D 采样向量传递给片段着色器。在片段着色器中，为立方体的每个部分着色，方法类似于将等距柱状投影图整齐地折叠到立方体的每个面一样。为了实现这一点，先获取片段的采样方向，这个方向是从立方体的局部坐标进行插值得到的，然后使用此方向向量和一些三角数学对柱状投影图进行采样，如同立方体图本身一样。对同一个立方体渲染六次，每次面对立方体的一个面，并用帧缓冲对象记录其结果就能将源DR图像转换为立方体贴图纹理。

# 4月14日 星期二

今日完成：渲染器搭建

今日实现间接漫反射的积分计算。主要目标是计算所有间接漫反射光的积分，其中光照的辐照度以环境立方体贴图的形式给出。给定任何方向向量wi，可以对预计算的辐照度图采样以获取方向wi的总漫反射辐照度。为了确定片段上间接漫反射光的数量（辐照度），我们获取以表面法线为中心的半球的总辐照度。为了生成辐照度贴图，我们需要将环境光照求卷积，转换为立方体贴图。假设对于每个片段，表面的半球朝向法向量N，对立方体贴图进行卷积等于计算朝向N的半球Ω中每个方向wi的总平均辐射率。求解积分需要我们在半球 Ω 内采集固定数量的离散样本并对其结果求平均值。分别给每个球坐标轴指定离散样本数量n1和n2以求其黎曼和。由于球的一般性质，当采样区域朝向中心顶部会聚时，天顶角 θ 变高，半球的离散采样区域变小。为了平衡较小的区域贡献度，我们使用sinθ来权衡区域贡献度。

指导教师评语

# 4月15日 星期三

今日完成：渲染器搭建

今日主要实现用户界面的交互。顶部菜单中，分为菜单项、功能解锁项和学习项。菜单项中有新建、打开和保存场景的功能按键；同时还有创建基础物体和导入模型的功能按键。在功能解锁项中，根据需求和设计将PBR功能划分为渐进式地3部分，PBR材质、间接漫反射和间接镜面反射；学习项将对应功能解锁项，提供开启每个部分文档的按钮。菜单栏中所有的按钮均用QAction实现。QAction类提供了抽象的用户界面action，这些action可以被放置在窗口部件中。Actions可以被添加到菜单和工具栏中，并且可以自动保持在菜单和工具栏中的同步。QAction可以包括一个图标、菜单文本、快捷键、状态文本，都可以在构造函数中设置。一旦QAction被创建了，那么就必须将它添加到相关的菜单和工具栏上，然后将它们链接到实现相应action功能的槽函数上。在PBR渲染器中将会用到QAction信号中的triggered信号。当点击界面菜单栏上的按钮时，对应的QAction会发出triggered()信号。在View类中，编写对应的槽函数，然后使用connect函数连接信号和槽。

# 4月16日 星期四

今日完成：渲染器搭建

今日实现渲染器用户界面交互。场景物体栏中将显示尝尽里的所有物体，物体将呈树形显示。同时，在场景物体栏还有设置天空盒和光源属性的菜单项。场景栏的实现使用的是Qt提供的QTreeView。它属于Qt4推出了一组新的ItemView类，它们使用Model/View结构来管理数据与表示层的关系。这种结构带来的功能上的分离给了开发人员更大的弹性来定制数据项的表示，它也提供一个标准的model接口，使得更多的数据源可以被这些ItemView使用。使用QTreeView的对应模型是 QStandardItemModel，这个是Qt对应ui界面最有用的模型，它可以用于树形控件、列表控件、表格控件等等和条目有关的控件。类QStandardItemModel负责保存数据，每个数据项被表示为类QStandardItem的对象。类QStandardItemModel将类QStandardItem表示的数据项组织起来，形成列表、表格、树甚至更复杂的数据结构，在本PBR渲染器中将数据项组织成为树形结构。该类提供了一组成员函数，向这些数据结构添加新的数据项，更改已经存在的数据项，或者删除已有的数据项。另一方面，作为一个模型类，它实现了QAbstractItemModel定义的接口函数，以使其他视图类能够访问模型中的数据项。PBR渲染器View类将调用类QStandardItemModel的成员函数appendRow()向某个树节点添加子节点，这样来实现对场景物体栏中实时添加物体项。

# 4月17日 星期五

今日完成：渲染器搭建

今日主要实现用户界面与底层类数据的连接。QStandardItem中的数据类型为QVariant。QVariant是一个最为普遍的Qt数据类型的联合，其用法与C++中union的用法差不太多。为了将View类中场景物体的显示能与Render类中场景物体连接起来，QStandardItem中的QVariant中将存储场景物体对应的指针。当用户点击场景栏中的物体时，可以获取对应的物体指针，通过这个指针就可以访问和操作该物体的属性。在场景物体栏种还能够进行天空盒和光源属性的设置，这个页面与场景物体页面以标签页的方式进行组织。这个标签页为Qt提供的QTabWidget组件。天空盒的设置为用户在用户界面中进行HDR文件的选择，然后View类将文件地址传递给Render类进行处理，具体实现函数为：AddEnviromentTexture。光源属性则是View类接收到数值改变的信号后，直接发送更改信号，将界面上设置的数值传递给Render类中的槽函数进行实时的修改。

# 4月20日 星期一

今日完成：渲染器搭建

今日主要实现用户界面的交互。属性栏将显示用户点击的物体对应的物体属性，属性的读取将通过点击的QStandardItem中存储的指针数据进行。属性栏的显示由Qt提供的QLineEdit、QSlider和QPushButton组件进行实现。对于每种贴图的添加，在添加后还需要在属性栏种显示贴图的预览。贴图的预览将用QLabel实现。在通过用户提供的地址读取了贴图图片后，用QPixmap对贴图数据进行放缩，使其缩小到预定好的QLabel大小，然后将QLabel的内容设定为处理好的QPixmap。菜单项全部是用QAction进行实现。所以需要响应用户操作触发的triggered()信号，以下为Operation类中对该信号响应的槽函数：AddSphere：用于响应在场景添加球体； AddCube：用于响应在场景添加立方体； AddModel：用于响应在场景加载用户指定的模型； UnlockMaterial：用于响应PBR材质功能的解锁； UnlockDiffuse：用于响应间接漫反射光照功能的解锁； UnlockSpecular：用于响应间接镜面反射光照功能的解锁；ShowPartOne：用于响应学习文档1的显示；ShowPartTwo：用于响应学习文档2的显示；ShowPartThree：用于响应学习文档3的显示。

# 4月21日 星期二

今日完成：渲染器搭建

今日对渲染器相机系统的实现。对于相机的旋转，将x轴、y轴和z轴的旋转角该称为3种欧拉角：俯仰角(Pitch)、偏航角(Yaw)和滚转角(Roll)。俯仰角是描述我们如何往上或往下看的角，偏航角表示我们往左和往右看的程度，滚转角代表我们如何翻滚摄像机。每个欧拉角都有一个值来表示，把三个角结合起来就能够计算在3D空间中任何的旋转向量。对于相机系统来说，只需要关心俯仰角和偏航角。根据鼠标的输入，计算当前帧与上一帧鼠标的偏移量。根据OpenGL的屏幕空间坐标系的排布，x轴的偏移量对应着偏航角的变化，y轴的偏移量对应着俯仰角的变化。将偏移量乘以一个灵敏度，就可以来进行相机的旋转调节。

# 4月22日 星期三

今日完成：渲染器搭建

今日实现间接镜面反射的计算。Cook-Torrance 镜面部分（乘以ks）在整个积分上不是常数，不仅受入射光方向影响，还受视角影响。如果试图解算所有入射光方向加所有可能的视角方向的积分，二者组合数会极其庞大，实时计算太昂贵。Epic Games 提出了一个解决方案，他们预计算镜面部分的卷积，为实时计算作了一些妥协，这种方案被称为分割求和近似法（split sum approximation）。 分割求和近似将方程的镜面部分分割成两个独立的部分，可以单独求卷积，然后在 PBR 着色器中求和，以用于间接镜面反射部分 IBL。分割求和近似法将镜面反射积分拆成两个独立的积分：一部分被称为预滤波环境贴图，它类似于辐照度图，是预先计算的环境卷积贴图，但这次考虑了粗糙度；第二部分等于镜面反射积分的 BRDF 部分。

# 4月23日 星期四

今日完成：渲染器搭建

今日实现间接镜面反射的计算。prefilter着色器中会根据粗糙度进行重要性采样。其实现方式为：开始一个大循环，生成一个随机（低差异）序列值，用该序列值在切线空间中生成样本向量，将样本向量变换到世界空间并对场景的辐射度采样。这里使用的是低差异 Hammersley 序列。基于特定的粗糙度输入和低差异序列值 Xi，获得了一个采样向量，该向量大体围绕着预估的微表面的半向量。输入的粗糙度随着预过滤的立方体贴图的 mipmap 级别变化（从0.0到1.0），我们根据据粗糙度预过滤环境贴图，把结果存在 prefilteredColor 里。再用 prefilteredColor 除以采样权重总和，其中对最终结果影响较小的采样最终权重也较小。此为之前所述渲染方程拆分的左半部分。brdf着色器中将会预计算BRDF。此为渲染方程拆分的右半部分，等同于在纯白的环境光或者辐射度恒定为1.0的设置下，对镜面 BRDF 求积分。着色器将角度和粗糙度作为输入，以重要性采样产生采样向量，在整个几何体上结合 BRDF 的菲涅耳项对向量进行处理，然后输出每个样本上F0的系数和偏差，最后取平均值。

# 4月24日 星期五

今日完成：渲染器搭建

为了以更有效的方式解决渲染方程中的积分，我们需要对其大部分结果进行预处理或称预计算。漫反射Lambert项是一个常数项（颜色 、折射率 和 在整个积分是常数），不依赖于任何积分变量。基于此，我们可以将常数项移出漫反射积分。由此，可以计算或预计算一个新的立方体贴图，它在每个采样方向中存储漫反射积分的结果，这些结果是通过卷积计算出来的。预计算的立方体贴图，在每个采样方向 上存储其积分结果，可以理解为场景中所有能够击中面向 的表面的间接漫反射光的预计算总和。

# 4月27日 星期一

今日完成：渲染器测试

功能测试主要对PBR渲染器按照需求分析中提出的功能需求进行测试与验证。点添加球体或是立方体会在场景中生成对应的物体并且初始的渲染方式为传统光照模式，同时还有不同种类的光源；在场景栏中，可以对物体的属性进行设置，同时也可以对光源属性进行设置；当点击开启贴图模式时，可以进入带贴图的渲染，需要提供贴图；点击模型加载可以将模型导入场景中，如图4-12所示；使用键盘上的WASDQE可以进行相机的前后左右上下的操作，并且使用鼠标可以进行视角的旋转；点击3份文档的打开按钮均能打开文档并且进行阅览。

指导教师评语

# 4月28日 星期二

今日完成：渲染器测试

性能测试主要对PBR渲染器按照需求分析中提出的功能需求进行测试与验证。

经测试，在生成基础物体时，PBR渲染器的响应时间未超过需求所规定的0.5s响应时间。在对模型加载的测试中，测试了三角形面片数为6954的模型，未出现卡顿现象，加载时间在0.5s以内；测试三角形面片数为19058的模型时，出现的卡顿现象，但响应时间满足需求中的性能要求；当测试更多三角形面片数的模型时，卡顿现象变得明显，响应时间也更长。

在帧数方面，当场景物体较少时，测试中的帧数可达到100以上，符合性能要求；但随着场景中物体的增多和PBR功能的开启，帧数会有所下降，但基本并没有低于60帧。

对于相机操作的响应，普通状态下响应时间正常；在模型加载的瞬间，测试时会有点卡顿的现象；对场景物体的属性改变，在测试中基本没有卡顿现象，响应时间完全符合需求中的性能要求。

# 4月29日 星期三

今日完成：论文写作

现在，可以开始编辑论文了。毕业论文表面看起来烦琐，其实还是一目了然的。大体上分为标题、中文摘要、英文摘要、目录、正文、致谢和参考文献以及附录等部分，标题又分为论文总标题、各部分的标题以及正文的三级结构标题，各个部分标题都有不同的格式与字体要求。在正文中，三级标题下面需要分开的。编辑的方法已经准备就绪了，下一步要进行的就是论文的内容了。这并不能急着动手编辑，因为编辑倒是个简单的问题，关键是要学习并掌握各种知识和原理，并在这个基础上设计出具有实际功能的实物来。而编辑，只是对这个过程进行具体记录、表达与总结，可见，下一步并不是动手写，而是疏通整个论文的各个知识要点，掌握其大概原理。

# 4月30日 星期四

今日完成：论文写作

我的毕业设计即将完成，回想这段时间，从拿到题目开始到设计完成，经历了很多困难和磨练，但总的来说，设计还是成功的，而且，在整个设计的努力过程中，我感觉自己在实践与动手能力方面得到了很大的提高，学会了怎样运用自己所学的知识去设计一个产品出来的整个过程。同时，也发现了很多的不足，给将来的工作与继续学习生涯敲响了警钟。现在，我就自己所做的这个设计作一个简单而又综合的介绍，让别人了解我做的这个产品到底是一个什么样的东西。

# 5月1日 星期五

今日完成：论文写作

今日写作设计部分。组件是模块化程序设计方法发展到一定阶段的产物，在软件工程的角度来考虑，开发者总是希望把一个庞大的应用程序划分成多个模块。其中，每个模块都保持一定的功能独立性，在协同共计良好的应用系统往往被切分成一些组件，这些组件可以单独开发、单独编译，甚至单独调试和测试。当所有的组件开发完成后，把他们组合在一起就得到了完整的应用系统。当系统的外作时，通过相互之间的接口来完成实际的任务。我们把每一个这样的模块称之为组件。其实，说得直白点，所谓组件，其实就是类，具有方法、属性和事件。

# 5月2日 星期六

今日完成：论文写作

今日编写论文实现部分。PBR渲染器使用C++进行界面与渲染部分的开发。界面部分使用的是Qt 5.12.6作为图形界面框架，该框架提供了许多方便实用的界面开发组件，提高开发效率。渲染部分则是使用OpenGL4.4，现在主流GPU对该版本也同样有所支持。代码编辑使用了Visual Studio2019，在其中安装了Qt项目支持的插件，让开发中能够兼容Qt的同时可以使用该IDE丰富的功能。在开发中，使用了glm数学库来辅助OpenGL中一些3D数学的数据结构的使用。在贴图读取方面，使用了stb\_image库来进行图片的读取，这个库支持了jpg、png等常见贴图文件读取的同时，还支持对环境贴图hdr文件的读取。在模型读取方面，使用了Assimp库来进行读取。

# 5月3日 星期日

今日完成：论文写作

今日进行实现部分的写作。相机为渲染场景中的观察者，它能够在场景中进行自由的移动与旋转来对场景进行观察。对于被渲染物体，它能够被用户进行位置、大小和朝向的设置。OBJ文件是一种在 3D软件模型之间的互导过程中是最不容易出错的一类标准3D模型文件格式，这得力于OBJ文件是一种文本文件。OBJ文件由一行行文本组成，每行由关键字开头来标记当前行的数据类型。OBJ文件不包含顶点的颜色信息，这些颜色信息将储存在一个后缀是".mtl"的独立文件中。mtl文件是obj文件附属的材质库文件，其中包含材质的漫射(diffuse)，环境(ambient)，光泽(specular)的RGB的定义值，以及反射(specularity)，折射(refraction)，透明度(transparency)等其它特征。由此可见，OBJ材质库的是不支持诸如金属度(Metallic)、粗糙度(Roughness)和AO等PBR材质的。

# 5月4日 星期一

今日完成：论文写作

今日完成论文的实现与后续部分。论文首先探讨了基于物理渲染的技术背景和PBR渲染器的实现意义进行了探讨，并对当前PBR技术的研究状况进行了调查。然后对PBR渲染器进行了需求分型并对渲染器系统进行了架构设计和模块划分。同时，还确定了开发与运行的环境。在实现部分，先对相关技术进行解释。讨论了微平面模型与影响这种表面光照表示的几个方程，与这种方程在实际运用种的近似。探讨了渲染方程的快速求解方法，对渲染方程的分割来进行预计算。然后讨论系统每个部分的实现工作。阐述了所用到的一些Qt组件与一些重要的数据结构。分析了整个渲染系统的实现思路与最终的实现效果。最后，跟据最开始的需求分析进行了系统的测试工作。对系统的功能测试介绍了测试任务与测试结果，并进行了分析。对系统的性能测试介绍了测试的方法与结果。最终，课题所要实现的PBR渲染器系统完成了实现并通过测试。

# 5月5日 星期二

今日完成：论文写作

通过长时间的努力，初稿完成，论文的撰写基本到了最后的工序，内容基本已经定型。初稿已经完成并交到了老师手里。然后老师批改后，给我们提出意见，进行进一步的修改，完成二稿。为了更好的进行论文撰写工作，在阅读完初稿的回复之后，要构思二稿的修改方式和内容。最后再在完成工作之后完成二稿的撰写。老师对初稿作出修改，提示论文观点论述存在的问题，调整文章布局，在行文、语言表达等提出观点。格式方面作出要求并提出修改的具体意见或建议。指导老师对我的初稿进行了详细的注释，并且提出了很多见解，大大提高了我对论文的认识。并通过进一步的翻阅文献，劲量的充实论文内容。针对搜集到的材料，进行悉心整理，根据自己拟定的提纲对收集到内容进行分类归纳，以便在撰写论文的过程中比较容易找到，运用到论文中去。为了让论文的表述更加清楚，也对论文的结构进行了整理，同时丰富了自己所表述的观点和内容。在构思论文的框架、行文及思想表达上更加完善。

# 5月6日 星期三

今日完成：论文写作

毕业论文的制作给了我难忘的回忆。在我徜徉书海查找资料的日子里，面对无数书本的罗列，最难忘的是每次找到资料时的激动和兴奋；为了论文我曾赶稿到深夜，但看着亲手打出的一字一句，心里满满的只有喜悦毫无疲惫。这段旅程看似荆棘密布，实则蕴藏着无尽的宝藏。我从资料的收集中，让我对我所学过的知识有所巩固和提高，在今后的日子里，我仍然要不断地充实自己，争取在所学领域有所作为。在这次毕业设计中也使我们的同学关系更进一步了，同学之间互相帮助，有什么不懂的大家在一起商量，听听不同的看法对我们更好的理解知识，所以在这里非常感谢帮助我的同学。在此更要感谢我的指导老师，是你们的细心指导和关怀，使我能够顺利的完成毕业论文。在我的学业和论文的研究工作中无不倾注着老师们辛勤的汗水和心血。老师的严谨治学态度、渊博的知识、无私的奉献精神使我深受启迪。在此我要向我的导师致以最衷心的感谢和深深的敬意。

指导教师评语