项目开发报告

**课 程 名 称**: C++综合实验

**课 程 代 码**: 190901029

**项 目 名 称**: GDI+开发基础——处理图形图像与文字

**年 级/专 业**: 2019级/计算机科学与技术

**学 生 姓 名**: 刘唐

**学 生 学 号**: 3120190971181

评 定 成 绩

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 平时表现  （20） | 开发能力  （30） | 项目工作量  （20） | 报告质量  （30） | 总 分  （100） |
|  |  |  |  |  |

教师签名： 20 年 月 日

**格式要求：**（成稿后删除本段及下文中的红色说明文字）

1. 不得更改版面格式如字体、字号、间距、页眉等（最好在本模板上修改）。
2. 中文字符数：4000 - 6000（审阅/字数统计）。
3. 内容以算法与技术方法为主，可以贴少量关键代码并解释，禁止大段代码。
4. **期末报告需交打印稿**（双面打印装订）和**PDF**稿，**非期末报告**只交**PDF**稿

**注意**：转成PDF后必须再仔细看一遍，是否存在各种转换错误**！**

GDI+开发基础——处理图形图像与文字

**摘 要**

GDI+（Graphics Device Interface plus）是Windows XP中的一个子系统，它主要负责在显示屏幕和打印设备输出有关信息，它是一组通过C++类实现的应用程序编程接口[1]。用GDI+处理图形图像与文字，是对GDI+的基础类与成员的了解以及入门，以便后续深入学习MFC与C++。

本设计学习了GDI+编程准备工作、实现了控制客户区属性的初始化、用Clear对客户区进行重绘、设置画图工具的属性、用“Draw-”函数进行绘画等。

本文对——————————————————————进行了分析和研究，重点介绍了————————————————————————————。

【**关键词**】GDI+;成员函数；填充；坐标系变换

**1绪论**

作为图形设备接口的GDI+使得应用程序开发人员在输出屏幕和打印机信息的时候无需考虑具体显示设备的细节,他们只需调用GDI+库输出的类的一些方法即可完成图形操作,真正的绘图工作由这些方法交给特定的设备驱动程序来完成,GDI+使得图形硬件和应用程序相互隔离.从而使开发人员编写设备无关的应用程序变得非常容易[2]。因此，GDI技术是一门需要重点掌握的技术。

**2 总体方案设计**

2.1 设计任务及要求

本次设计的必要功能要求：

1. 程序UI：以MFC对话框形式呈现，主要功能入口为菜单栏，客户区为除窗口及菜单栏其余部分，为矩形。
2. 绘制图形：画线:从直线到笛卡尔曲线、从矩形到多边形；填充：矩形，椭圆，多边形；叠加：渐变以及网状线。
3. 绘制文字：绘制，图案，纹理，渐变。
4. 坐标系变换：平移、旋转、缩放、镜像。

2.2 总体方案设计

基于GDI+技术，通过MFC以及C++实现。在Windows 10、 Visual Studio 2019的环境下调试编译运行甚至发布。

**3 详细设计**

在这一章，我将场景渲染和UI交互两大功能的原理和具体实现方法一一呈现出来，并给出部分流程图和关键代码。

3.1 场景渲染详细设计

3.1.1 灯光布置

无论是现实生活还是虚拟世界，灯光的作用都是照亮场景[8]。适当运用灯光不仅照亮场景，还会增加场景气氛，改变物体的材质效果。Unity 3D引擎的灯光主要分为平行光（Directional light）、点光（Point light）、聚光（Spotlight）和面片光（Area Light）四种[6]（图3.1）。

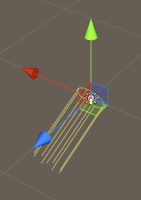
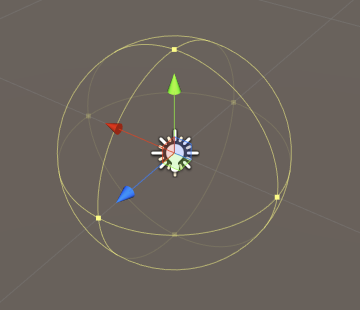
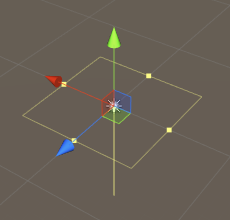
   

图3.1 四种光源类型

经过分析三维场景的房间模型后，为了能获得更加真实的光线效果，主要采用了平行光、点光和面片光三种光源，获得不同的渲染效果。

最终灯光布置完成如图3.2所示

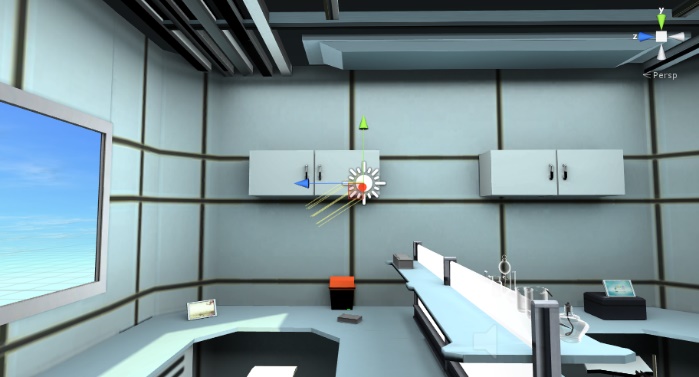


图3.2 实验室内部的灯光布置

3.1.2 光线烘焙

光线烘焙，是将场景中的静态物体基于光线计算得到贴图，并将此贴图迭加在这些静物表面上，显示出更为逼真的光线效果。实验室墙壁、天花板、桌面、其它不动的物品都属于静物，因此，采用光线烘焙的方式能够获得更好的效果。光线烘焙的主要内容：选取烘焙物体、设置灯光参数、设置烘焙参数。

从菜单中Window - Lighting进入Unity3D集成的Lightmapping窗口，如图3.3所示，烘焙全局光照的主要调节参数：

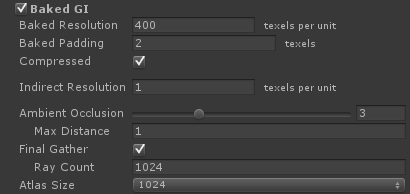


图3.3 光线烘焙主要参数

烘焙结束后，资源目录会自动生成名为Light的文件夹，用于存放若干张的光线贴图。此时，静态物体的表面贴图已经被光线贴图所替代，原来比较生硬的光线已经变得比较柔和，更符合真实条件下的光照效果，显得更为逼真。如图3.4所示，室内光线在烘焙前后已经发生了根本性的改变。烘焙前的阴影边缘锐度比较高，阴影颜色很深；而烘焙后的阴影边缘锐度比较低，阴影颜色比较浅。同时，桌子表面和桌子上物品的边缘部分都出现了非常漂亮的漫反射阴影，为整个场景增添了温暖的感觉。

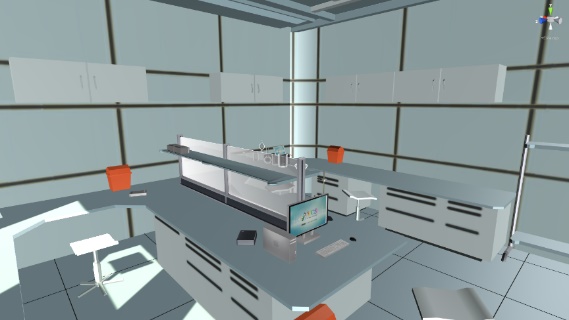
 

图3.4 烘焙前（左）和烘焙后（右）效果对比

3.2 UI详细设计

3.2.1首页菜单

首页是启动程序后，用户能够自主操作的第一个页面。因此，首页不但要引导用户进入主页面，而且还要展示出程序的特色。

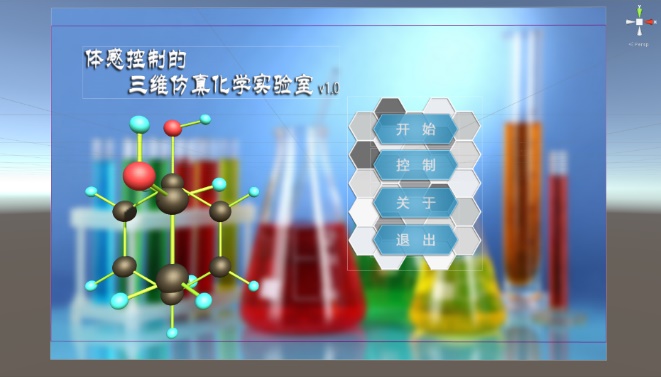


图 3.5 首页UI设计

最终首页的设计效果如图3.5所示。在页面的上半部分，是本设计题目。在页面的左半部分，有两个三维分子模型，分别是苯酚（C6H5OH）和乙醇（C2H5OH）。在进入程序时，会随机展示其中的一个三维分子模型且以较慢的速度匀速自传。如果用鼠标对其进行拖动，则三维分子模型会随着鼠标的拖动而转动。

3.2.2设置菜单

考虑到用户电脑存在性能差异，低配置电脑在运行高画质效果时可能会存在不流畅的问题。因此，本设计增加一个设置菜单，为用户提供调节画质和音量的两个主要功能，方便用户调节音量大小和画质精细度。为了美观，为设置菜单的收、放添加了动画表现，实现平滑过渡。最终设计如图3.6所示，整体风格以六边形为主，增加了音量旋钮、画质调节以及关于按钮。

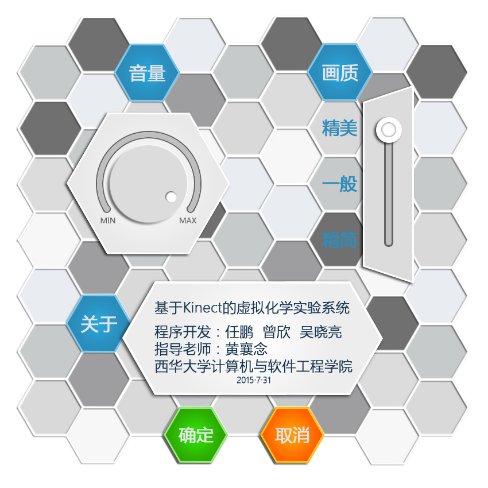


图3.6 设置菜单【注意图和图题不能分页】

3.2.3 控制面板

开始

固体

展开固体

药品菜单

展开液体

药品菜单

展开气体

药品菜单

液体

气体

清除

设置

退出

展开系统

设置菜单

退出出？

清除全部

清除单个

收缩菜单

点击图标

展开菜单

结束

是

否

图3.7“控制面板”的控制流程图

正确画法：方框为加工处理框，需要动词。“清除”下面应先判断再分支处理

根据六边形设计，考虑画面的协调性，为控制面板设计了6个二级菜单，图3.8所示。

图3.8 控制面板收缩（左）和展开（右）状态【注意：图和图题不能分页】

默认情况下，二级菜单处于收缩状态且不能点击。点击中部Logo图案后展开全部二级菜单。“固体”、“液体”、“气体”按钮分别提供了本次实验所需要的实验药品，点击可展开具体的实验药品。

3.2.4 Kinect与鼠标控制

Unity引擎没有控制鼠标位置和鼠标事件的API，在Window平台下我们借用系统API来达到目的。Windows系统库 “user32.dll”包含用户界面相关应用程序接口，用于包括Windows处理、基本用户界面等如创建窗口和发送消息，其中就有我们需要的鼠标事件处理。

C# 使用Dllmport 属性需要将它放在导出函数上，传入参数需要源dll中的名字。代码中mouse\_event和SetCursorPos两个方法分别用于模拟鼠标事件和设置光标位置，且使用方法组合还可以模拟如鼠标移动、拖动的效果。

代码如下所示：

[DllImport("user32.dll")]

private static extern void mouse\_event(MouseFlags dwFlags, int dx, int dy, int dwData, System.UIntPtr dwExtraInfo);

[DllImport("user32.dll")]

public static extern int SetCursorPos(int x, int y);

Kinect开发SDK已经实现了部分默认手势动作识别[12]。支持的手势如下：

* + - 举起左手/举起右手：手举过头顶超过1s。
    - 停止：双手放在腰部以下。
    - 挥动：左右手在左右方向上交叉挥动。
    - 滑动到左边：右手滑动到左边。
    - 滑动到右边：左手滑动到右边。
    - 放大：左右手在一起，在肘部上方，然后两手向不同方向移动。
    - 缩小：左右手至少距离0.7m，然后双手靠近对方。

使用系统定义的手势流程如图3.9。

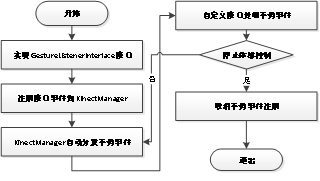


图3.9 Kinect手势控制流程【不能贴图，要求绘制才更清晰】

其中，GestureListenerInterface主要的方法为 GestureInProgress()和 GestureCompleted()，分别在手势处理过程和手势完成后自动调用，通过实现该接口并重写这两个方法来达到感知用户动作信息的目的。

3.2.5 物体移动

为实现鼠标点击物体并拖动，需要使用从摄像机发出的穿过鼠标的屏幕射线，检测鼠标指向了哪个物体，从而对指向的物体进行操作。Unity物理引擎为我们提供了一个射线类Physics.Raycast，它可以发射出符合需求的射线，并将射中的物体信息返回回来。

以下代码展示了如何使用屏幕射线检测鼠标指向的物体。

// 用来检测物体的屏幕射线

Ray ray = GetComponent<Camera>().ScreenPointToRay(Input.mousePosition);

// 射线射中的物体对象

RaycastHit hit;

// 调试环境中显示射线，运行环境不可见

Debug.DrawRay(ray.origin, ray.direction \* RayLength, Color.yellow);

// 发射射线并返回是否射中物体

bool isHit = Physics.Raycast(ray, out hit, RayLength, mask.value);

3.2.6 物体吸附

物体发生碰撞的必要条件：两个物体都必须带有碰撞器(Collider)，其中一个必须带有Rigidbody刚体。引擎能检测碰撞方式：碰撞器和用触发器。

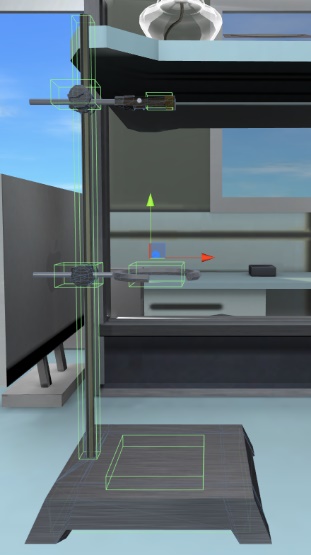


图3.10 铁架台包含的触发器

如图3.10，为“铁架台”关键部位添加了触发器（绿色线条状立方体），当其他物体靠近时，就能感知并自动吸附到预设的位置上，使化学实验装置连接更加紧密。

通过使用触发器，就能很方便的捕获两个物体发生碰撞，并自动回调触发信息检测函数如MonoBehaviour.OnTriggerEnter(Collider collider)在进入触发器时回调，MonoBehaviour.OnTriggerExit(Collider collider) 则在退出触发器时回调，最终实现吸附物体的效果。

3.2.7 气体模拟

Unity引擎的粒子系统可以模拟烟雾、气流、火焰和各种大气效果。调节参数可以获得不同的粒子效果如颜色、半径、发射角、速度等，“氯气”效果见图3.11所示。



图3.11模拟“氯气”效果

**4 结论**

随着虚拟现实技术的发展，现在已经广泛运用在医学、娱乐、游戏、仿真、教育等领域，人们越来越真实的体验到虚拟世界的精彩。如今，国内各个中学的化学实验室都比较完善，能完成课本上的大部分化学实验。由于某些化学实验受到高危险、高污染、高成本等因素的限制，使得这些化学实验往往只能通过书本或互联网进行了解，大大限制了学生对这些化学反应实质上的理解程度。因此，通过使用虚拟现实技术真实地模拟化学反应的整个过程，让学生参与整个实验操作，能有效地提高学生对这部分化学知识的理解程度。

本系统采用基于计算机的数字化三维技术，逼真地模拟化学场景、实验器皿、化学药品、化学反应过程等，并采用基于传统操作方式的键鼠操作和基于未来人机接口技术的微软Kinect体感设备来进行人机交互。

**参考文献 (数量≥5，注意写法格式)**

[1] 百度百科.GDI+[OL]. https://baike.baidu.com/item/GDI%2B/2128162

[2] CSDN.GDI编程小结[OL]. https://blog.csdn.net/byxdaz/article/details/5949590

[3] 金玺曾.Unity 3D手机游戏开发[M].北京:清华大学出版社.2013.8

[4] 宣雨松.Unity 3D游戏开发[M].北京:人民邮电出版社.2012.6

[5] 百度百科.游戏引擎[OL].http://baike.baidu.com/view/33343.htm

[6] 百度百科.Unity 3D[OL].http://baike.baidu.com/view/2299410.htm

[7] Art Eyes.创意UI：Photoshop玩转移动UI设计[M].北京:人民邮电出版社. 2015.1

[8] Tynan Sylvester.体验引擎：游戏设计全景探秘[M].北京:电子工业出版社. 2015.3

[9] Jon Skeet.深入理解C#（第3版）[M].北京:人民邮电出版社. 2014.4

[10] 时代印象.3ds Max 2012基础培训教程[M].北京:人民邮电出版社. 2012.6

[11] 时代印象.中文版Maya 2012实用教程[M].北京:人民邮电出版社. 2012.4

[12] 余涛.Kinect应用开发实战：用最自然的方式与机器对话[M].北京:机械工业出版社.2012.11

[13] 王森.Kinect体感程序设计入门[M].北京:科学出版社. 2014.12