学号	姓名	论文规范性（10）	问题分析与调研（30）	方案创新性（20）	实验结果分析与讨论（40）	结课论文总成绩（100）
21301098	廖思瑜	7	22	16	34	79

文字间很多空格

计算机图形学课程论文

题目：实时渲染和计算机动画在游戏中的应用

学　　号：21301098

姓　　名：廖思瑜

学　　院：软件学院

专业班级：软件 2104

目录

摘要

游戏制作中，涉及了大量实时渲染和计算机动画的技术，随着科技的发展，高质量游戏不断涌现，它们都在实时渲染和计算机动画方面表现十分出色。本文将介绍现有技术的发展现状，学习优秀的渲染和动画作品，并对技术进行应用，完成一个 3D 贪吃蛇，包括经典贪吃蛇的功能以及相机视角转换，光线追踪等等。

1 引言

如今实时渲染技术和计算机动画技术发展的越来越成熟，也越来越多人投入到他的研究和应用中，其中比较热门的应用就是游戏的开发和设计。一个好的游戏少不了优秀的实时渲染技术和优质的计算机动画。作为学生，我们可以尝试开发简单的小游戏，应用实时渲染和计算机动画技术。

2 相关工作介绍

在具体的游戏开发中，主要涉及的计算机图形学领域为实时渲染和计算机动画，下面分别介绍两者研究现状及本次实验的选择方向和理由。

2.1 实时渲染研究现状

实时渲染技术作为计算机图形领域重要的一环，被广泛运用在游戏，动画之中。随着科技的发展，也被用于类似 VR 等增强现实应用中。它通过实时的创建画面样式，动态效果等等，将实时变换的画面呈现给用户。

今年三月份，题目为无尽阶梯的第八届世界渲染大赛落下帷幕，展示了全球渲染技术领域的最新发展和创新。在本次比赛中，来自各个国家和地区的渲染技术专家和研究者展示了他们的技术成果和创意作品，涵盖了光线追踪、光栅化、材质渲染、渲染优化等多个方面。这些作品不仅在视觉上令人惊叹，而且在技术层面上也展示了对实时渲染的深入理解和创新应用。

3D 渲染的两种主要类型是光线追踪和光栅化。光线追踪考虑了场景中的光路，从而产生更真实的图像。然而，它的计算成本很高，因此不太适合实时渲染。另一方面，光栅化速度更快，因为它将场景划分为 2D 图像，然后由 GPU 进行处理。该技术更适合实时渲染，因为它生成图像的速度更快[1]。

用于实时渲染的其他流行技术是储层采样和重采样重要性采样。这些技术在对直接光照

进行采样时非常有用，因为它们有助于更快地生成图像，而不会影响图像质量。此外，结合反射和阴影等高频细节可以提高图像的真实感，这在游戏开发中至关重要。

2.2 计算机动画

计算机动画在游戏中的应用同样非常平凡，它涉及到很多个学科，涉及到的领域很宽泛，它除了需要使用计算机图形学相关的技术，还有数字图像处理，运动控制，美术建筑等等。

计算机动画的应用主要体现在五个方面,应用最广泛的是娱乐用途,也就是常说的动画与游戏,除此之外,计算机动画在宣传方面也有着很多的应用,不仅仅包括广告,而是指包括能 用动画形象的表示某种信息的宣传。在国内,国际的各方面的形势变化如城市规划等工作,也 是非常适合用计算机动画来表现的领域,在仿真模拟方面,计算机动画的可以通过动画的形式 仿真而应用于军事,医学等领域,而在教育方面,采用计算机动画的教学方式更加生动与直 接,可以充分调动学生的积极性,产生良好的教学效果,另外,计算机动画相对手工操作是更 为高效的特点让它可以应用于多种领域,产生巨大的经济效益与社会效益[2]。

计算机动画的应用在游戏中的应用主要有：角色建模与动画，场景设计与特效，物理模拟与交互，剧情表现与情节推进，用户界面与交互设计。

2.3 实验所选择的技术方向及理由

本次选择方向为应用实时渲染和计算机动画技术，制作一个简单的 3D 贪吃蛇，实现对游戏角色的移动控制，页面的实时渲染以及游戏中触发各种条件的判断。选择该方向的原因主要是对游戏领域较为感兴趣，希望探索其背后的原理以及开发的流程，并且做出简单的应用。

3 方法描述

3.1 实验过程

1) 数据的收集

实验前需要收集一些纹理贴图数据，obj 模型等，这里选用了实验一中已经制作好的水壶模型和甜甜圈模型作为游戏中贪吃蛇的食物。另外贪吃蛇由立方体作为头部，圆形作为蛇身形成。地板的贴图使用了一个木桌子的纹理

2) 实验环境设置

在 windows 操作系统下开发，使用了 glut 库

3) 实验设计

实验计划先进行地图的生成，之后是模型的加载和贴图，最后加入游戏各种判断逻辑，监控用户操作，完成交互。

3.2 研究方法

1) 技术和工具

使用 C++作为开发语言，使用 Visual Studio 2022 作为开发工具

2) 开发过程

明确实验需求，配置实验环境，按需求进行开发。

4 实验设置

4.1 实验环境

主要采用 C++作为开发语言，使用 Visual Studio 2022 作为主要的开发工具，

在图形库和渲染工具方面，本文采用 OpenGL 作为核心图形库。此外，使用 GLUT 库处理 OpenGL 窗口系统之外的所有操作。自定义 LoadTexbmp.h 库来管理游戏中的纹理资源，包括食物、蛇身等元素的贴图加载和渲染，确保游戏在视觉效果和操作流畅性上达到预

期。

4.2 实验步骤

1) 步骤一：部署开发环境

在 Visual Studio 2022 中创建新的 C++项目。配置项目以使用 OpenGL 和 GLUT 库

设置项目的基本结构。

2) 步骤二：创建 3D 场景和基本地图对象

```
float map_witdh = 100;//地图的宽度
float map_depth = 100;//地图的深度
```

3) 步骤三：完成蛇相关的功能

a. 定义“蛇”节点的数据结构，包括关节位置，方向

```
typedef struct _SnakeNode {
    float mposx;//关节位置
    float mposz;//
    float mDriection[2];//方向
    float mAngle;//
}SnakeNode;
```

b. 定义蛇是否碰到食物，是否自己碰撞，方向改变，关节数量改变等函数。

```
class Snake
{
public:
    Snake();
    ~Snake();
    void  ChangeDirection(int diretion,int mode =0);//改变m_MoveDriection方向，和蛇身体方向做判断
    void  MoveForWard(float t);//移动 -->计算方向 -->改变位置
    void  Draw();//绘制 注意头和尾部的区别
    bool  CollisonFood(float foodx, float foody);//检测是否吃到食物
    std::vector<SnakeNode> m_snake;//关节
    float m_MoveAng;//蛇头移动方向
    float m_MoveSpeed;//蛇头移动速度
private:



    float m_MoveMap[2];//活动范围
    void  Init();//初始化
    void  CollisonSelf();//自己是否碰撞
};
```

c. 蛇身各个节点衔接

```
glPushMatrix();
if (ptr == m_snake.begin())
{
if ((*ptr).mposx*2.0 > m_MoveMap[0])x = -m_MoveMap[0];
if ((*ptr).mposx*2.0 < -m_MoveMap[0])x = m_MoveMap[0];

    if ((*ptr).mposz*2.0 > m_MoveMap[1]) y = -m_MoveMap[0];
    if ((*ptr).mposz*2.0 < -m_MoveMap[1])y = m_MoveMap[0];
}
(*ptr).mposx += x;
(*ptr).mposz += y;
glTranslatef((*ptr).mposx , 0, (*ptr).mposz);
glRotatef((*ptr).mAngle, 0, 1, 0);
```

4) 步骤四：添加游戏逻辑和交互

监听用户键盘操作，并实现对变量的对应变换

a. 蛇的移动变换

```
void KeyboardEvent(unsigned char key, int x, int y)
{
    switch (key)
    {
    case 'W'://上移动
        m_snake.MoveForWard(0.1);
        break;
    case 'S'://下移动
        m_tranlate[1] -= 0.1;
        break;
    case 'A'://左移动
        m_snake.ChangeDirection(-1, 1);
        break;
    case 'D'://右移动
        m_snake.ChangeDirection(1, 1);
        break;
    }
}
```

b. 视角的旋转变换

```
void MotionEvent(int x, int y)
{
    if (m_bMouseDown)//如果鼠标左键被按下
    {
        m_rorate[0] += y - m_MouseDownPT[1];//通过滑动鼠标改变旋转的角度
        m_rorate[1] += x - m_MouseDownPT[0];//通过滑动鼠标改变旋转的角度

        m_MouseDownPT[0] = x;//记录当前X坐标
        m_MouseDownPT[1] = y;//记录当前Y坐标
    }
}
```

c. 蛇碰撞自身的检测

```
void  Snake::CollisonSelf()//自己是否碰撞
{

    for (std::vector<SnakeNode>::iterator ptr = m_snake.begin(); ptr != m_snake.end(); ptr++)
    {
        if (ptr != m_snake.begin())
        {
            //判断 头部和关节的距离
            float dis = (sqrt((*ptr).mposx - (*m_snake.begin()).mposx)*((*ptr).mposx -
                (*m_snake.begin()).mposx) + ((*ptr).mposz - (*m_snake.begin()).mposz) *((*ptr).mposz - (*m_snake.begin()).mposz));
            if (dis < 0.8)
                Init();
        }

    }

}
```

d. 自动生成食物的逻辑

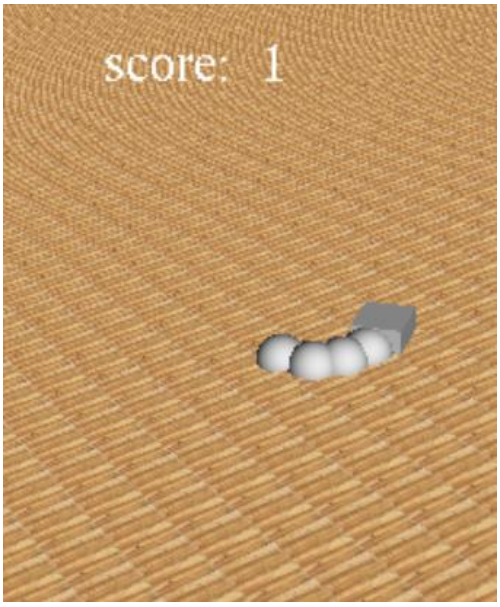
```
void Food::Init()
{
    m_food.clear();
    m_count = 10;
    m_MoveMap[0] = 70;
    m_MoveMap[1] = 70;
    for (int i = 0; i < m_count; i++)
    {
        SinglFood m_SinglFood;
        m_SinglFood.mposx =  rand() / (float)RAND_MAX * m_MoveMap[0] - m_MoveMap[0] / 2;
        m_SinglFood.mposz =  rand()/ (float)RAND_MAX * m_MoveMap[1] - m_MoveMap[1] / 2;
        m_SinglFood.mode = rand() / (float)RAND_MAX *2.9;
        m_food.push_back(m_SinglFood);
    }
}
```

5 实验结果与分析

5.1 实验结果展示

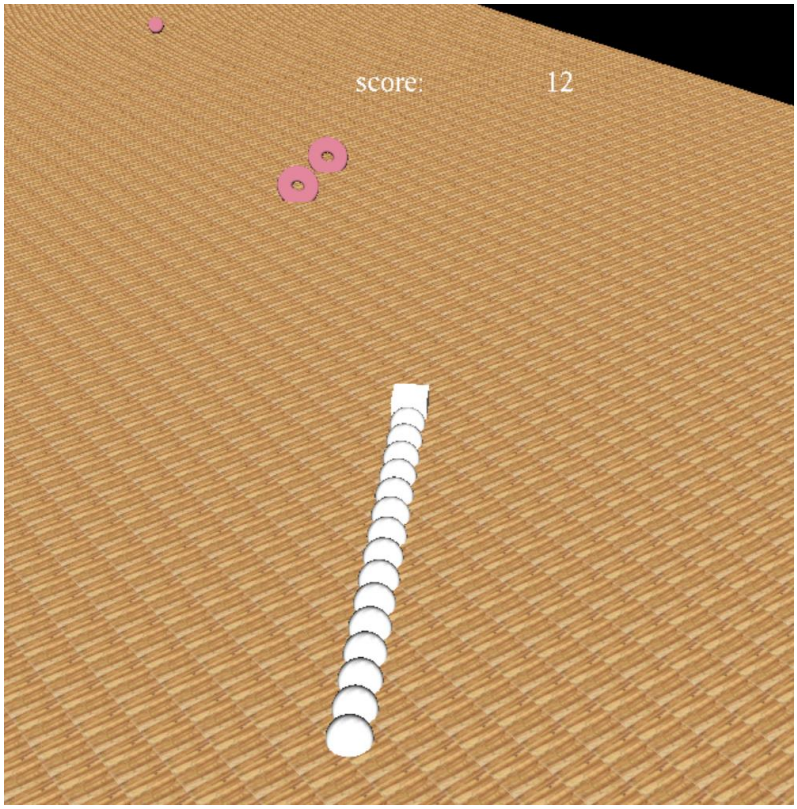
1) 蛇的移动和加速：

按 A，D 或者左右键可以让蛇进行转弯，加速的功能比较难展示，玩家通过按上建使蛇加速。

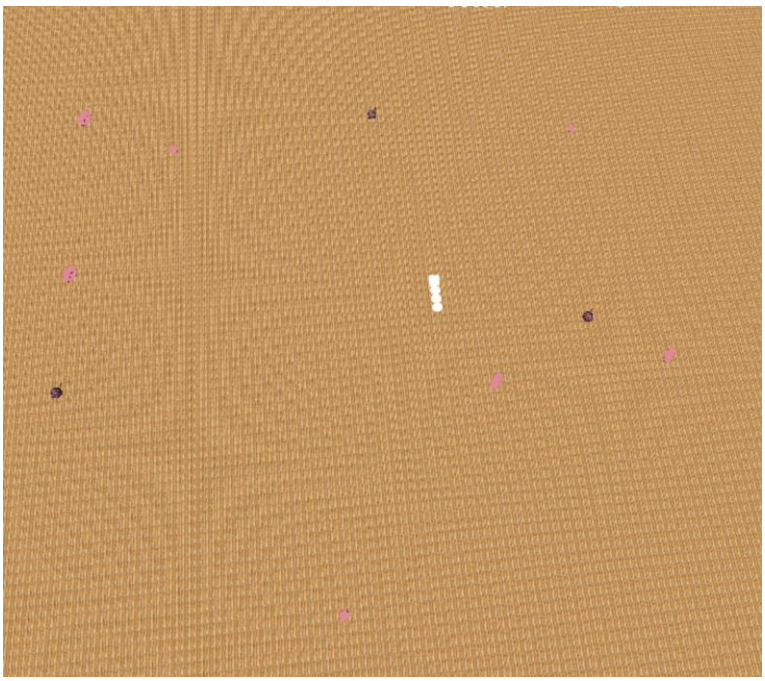


2) 蛇吃到食物

蛇迟到食物 score 会加一，同时蛇的长度会增长一个节点。



3) 食物不断自动生成



4) 蛇碰撞地图边缘或头碰到尾，游戏失败

5.2 实验结果分析

最终能实现一个简单的贪吃蛇游戏，包括吃食物蛇对应变长，碰到边缘或者头部碰到自身就结束游戏，另外还支持视角转换，视角旋转，放大缩小，在不同的距离下可以发现不同光影效果的蛇主体。

6 结论

6.1 总结研究的主要内容和结论

本次实验运用了简单的实时渲染，光线追踪，使得贪吃蛇在不同角度，距离，看起来的颜色和所展示的效果有所不同。以及动态生成物体，增加游戏趣味性。实验过程中对于贪吃蛇各个节点如何衔接尝试了多种考虑，最后才选出一种合适的方式，但仍有可以改进的空间，需要解决蛇身变长一段距离后，整条蛇无论如何转弯都只能作为一个大圆的一段圆弧，不再有头碰到身子的可能。

6.2 探讨未来可能的研究方向

未来可能继续尝试使用计算机图形学课程的相关技术，尝试更多可能性，也尝试更复杂的游戏逻辑。

7 参考文献

[1] 《有哪些流行的实时渲染技术？深入了解各种实时渲染技术》——3D 渲染技巧分析

[2] 《电子技术与软件工程》2016 年 22 期 ——王佳隽