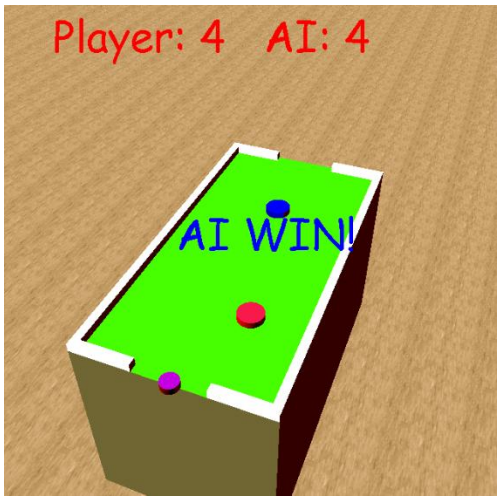
计算机实时动画大作业报告

2012011307 黄必胜

2012011314 鲁逸沁

1 选题与环境配置

选题： Air Hockey Game(Project1)
编程系统：Win7
编程语言：C++
编程工具：Visual Studio2010，OpenGL，GLUT，GLAUX
VS2010 中的额外编译选项：
配置属性->常规->项目默认值->MFC 的使用，置为“在共享 DLL 中使用 MFC”
最终效果图：



Demo 程序及视频均置于 bin 目录下。

2 功能需求与实现

Task1：实现 3D 对象的绘制(20 points)

功能需求：

需要绘制以下 3D 对象 Floor，Table，Wall，Puck 和 Mallet，以构成 Air Hockey Game 的界面部分。以下分别对其实现进行讨论。

Floor：

构成：一个正方形面片。重点在于载入指定的纹理。

载入纹理的方法：

1) 调用 **auxDIBImageLoadA()**返回 **AUX_RGBImageRec** 指针。
2) **glTexImage2D()**将载入的图像与纹理对象绑定。
3) 在 OpenGL 初始化函数中，参数选择为，线性滤波，周期重复。
4) 并启用纹理 **glEnable(GL_TEXTURE_2D)**。

绘制函数：**glBegin(GL_QUADS)**。

代码位置：函数 **drawFloor()**。

Table, Wall：

Table 分为两部分，底座和桌面。

底座构成： 一个 Cube，置于 Floor 之上。

参数： Cube 的长宽高置为(2.2，2.4，4.2)。

绘制函数： **glutSolidCube()**。

桌面构成：一个四边形面片，覆盖在底座的上面。

参数：颜色置为绿色。

绘制函数： **glBegin(GL_QUADS)**

Wall 分为左边界、右边界和前后分别的两个边界（形成球门）。

构成： 一个 Cube，置于桌面对应的位置。

参数： 边界的宽度和高度为 0.1，长度与桌面相同。

绘制函数： **glutSolidCube()**。

代码位置： **DrawTable()**和 **Drawedges()**。

Puck，Mallet：

构成： 一个圆环状加顶部的一个圆面片，呈饼状结构，

参数： Puck 半径 0.1，高度 0.08

Mallet 半径 0.15，高度 0.08

绘制函数： **gluCylinder()**和 **glBegin(GL_POLYGON)**。

Task2：实现 Puck 的运动(30 points)

功能需求：

游戏中需要实现 Puck 在桌面上的匀速运动，包括与墙面和 Mallet 的碰撞效果。

实现：

维护 Puck 的速度和位置，并让其在固定的时间间隔沿着速度的方向将位置移动固定的步长，并注意时刻保持将 Puck 限制在桌面之中。固定的时间间隔可以利用计时器来完成。计时器调用函数：**glutTimerFunc()。**

实现 Puck 的运动核心是在恰当的时机改变 Puck 的运动方向。运动方向的改变与墙面或 Mallet 的碰撞有关。进行碰撞检测的方法是：在每一个绘制的瞬间，检测 Puck 的位置与墙

面和 Mallet 位置的关系。分为以下几种情况。

1) 无碰撞。
2) 与墙面的碰撞。
   条件：Puck 的中心位置与边界的距离小于 Puck 的半径。
   效果：改变 Puck 的垂直于墙面的运动方向。
3) 与 Mallet 的碰撞检测。
   条件：Puck 的中心位置与 Mallet 的中心位置距离小于它们的半径和。
   效果：将 Puck 的运动方向置为从 Mallet 中心到 Puck 中心的径向量。

Task3：实现玩家控制和简单对手 AI(30 points)

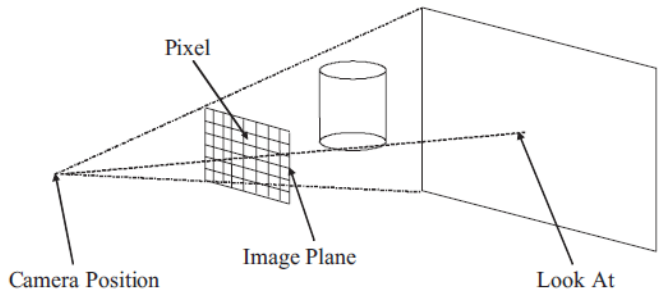
功能需求：

玩家应该通过鼠标控制己方 Mallet 的位置，并需要与简单的对手 AI 进行对战。

实现：

玩家控制，最大难点在于找到鼠标从屏幕坐标到世界坐标的映射关系，需要计算鼠标的指向在世界坐标系中的位置，并把玩家的 Mallet 绘制在正确的位置。这同时也是本项目中的最难点。以下是解决该问题的办法：

1) 确定视点，视方向和视角。例如从以下函数可知，视点在向量 eye 的位置，视方向在 focus 向量的位置。视角为 fovy，程序中设置为 45°。
   gluPerspective(fovy,(GLfloat)width/(GLfloat)height,0.1f,100.0f);
   gluLookAt(eye.x, eye.y, eye.z, focus_x, focus_y, focus_z, 0, 1, 0);
2) 将模型类比于人的眼睛。视点 camera 是眼睛内部的焦点，而屏幕相当于视网膜，在世界坐标系中相当于一块四边形面片。可以通过计算寻找到 camera 和 retina 在世界坐标系中的位置。程序中假设 retina 与 camera 的距离为 1。retina 相当于下图中的 Image。



3) 当鼠标指向屏幕上的某点时，首先在世界坐标系中找到 retina 上的对应点。引一条从 camera 到 retina 的射线，并延长至与桌面相交。交点即为鼠标所指向的世界坐标系的位置。建立 retina 模型和找到其与桌面的交点都需要较多的数学计算。

简单对手 AI 的编写，策略是保持在一条横线上进行移动。保持纵坐标不变，以一定的速度改变横坐标，尽量跟随着 Puck 的位置以期将其反弹。在实际时间中，将对手 AI 移动的速度设置到了合适的大小，以保证游戏具有一定的可玩性。

Task4：实现视角变换，游戏流程和拓展功能(20 points)

功能需求：

游戏应该能实现从至少两个不同的视角进行游戏，应该能遵照正常的游戏流程，提示胜负关系，并实现一些拓展功能。在本项目中我们实现了计分榜，多视角转换的拓展功能，在物体外表的显示上也有所优化。

实现：

视角变换，其核心难点已经在上一个 Task 中解决了，所以只需要改变视点和视方向就可以实现在不同的视角观察游戏。

游戏流程，其实现通过检测 Puck 的位置是否在球门内部。当 Puck 处于某一方的球门的时候停止 Puck 的运动，并提示另一方胜利。

拓展功能 1：计分榜

功能描述：在屏幕上显示多个回合的比赛过后双方的积分。

实现：程序记录双方的积分值，并在屏幕的左上角实时绘制显示。

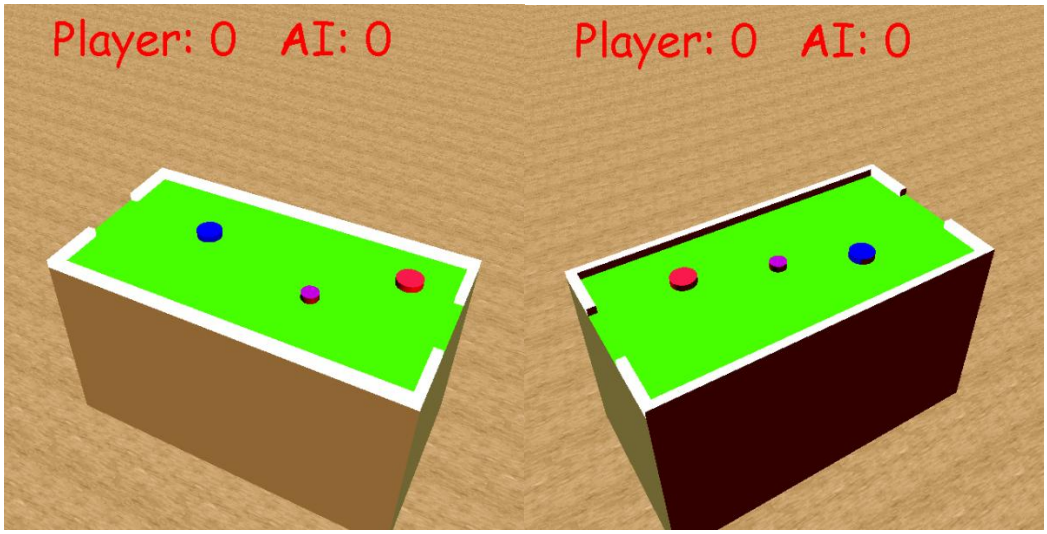
绘制相关函数：wglUseFontBitmaps()，glCallList()，并通过 WINAPI 设置字体。

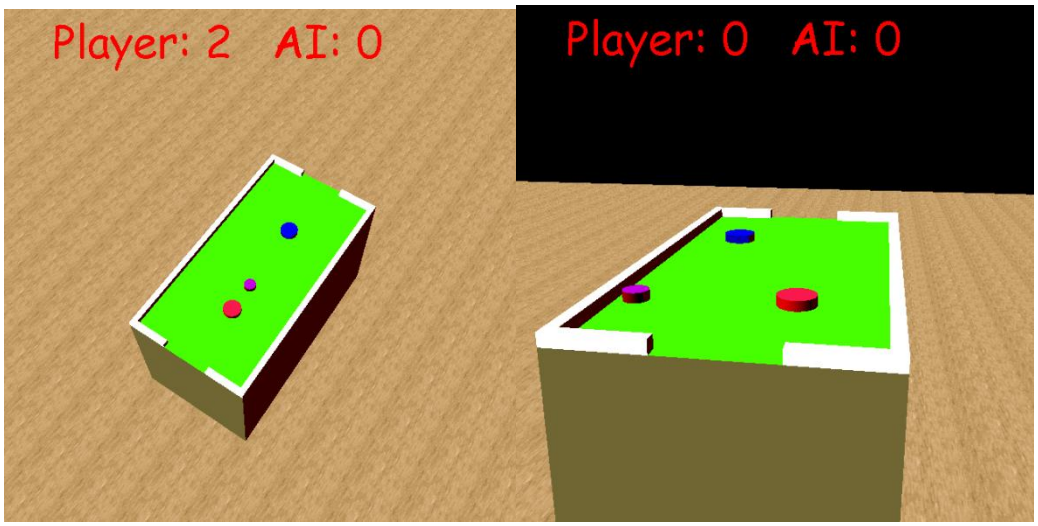
拓展功能 2：多角度变换

功能描述：原有的作业要求只要求两个角度的变换，而在我们的程序中实现了水平方向和垂直方向的视角自由移动。实现难点在于保持鼠标能够有效地控制 Mallet 的位置。

实现：进一步改变视点和视方向的位置即可。

效果图：





拓展功能 3：利用光照模型改善物体的外表。
实现：程序中设置了绘制物体的材料模型，以获得更好的物体显示效果。

3 使用说明

bin 目录下的 demo.exe 是相应的可执行文件。
按键说明：
方向键，进行视角的变换。
Enter 键进行新一轮的比赛。
鼠标移动，控制玩家的 Mallet。