**《计算机图形学》课程项目说明文档**

****

**项目名称： 冰壶游戏**

**指导教师： 童若锋**

**小组成员： 朱奕杰 韩闰罡**

**蒋弘毅 诸嘉涵**

**冰壶游戏说明文档**

1. **基本信息与游戏概述**
   1. **开发环境**

|  |  |
| --- | --- |
| 计算机 | CPU: ≥2.0GHz 内存≥2.0GB |
| 操作系统 | Windows10 |
| 开发工具 | VS2013, 3DS MAX |
| 开发语言 | C++, OpenGL |
| 办公软件 | Microsoft Office2007/2010系列产品 |

* 1. **游戏介绍**

“冰壶游戏”是一款基于真实体育竞技比赛的改编游戏。从1998年开始，冰上溜石（冰壶）被列为冬奥会正式比赛项目，比赛是以队为单位在冰上进行的，并设有冰壶世锦赛。本游戏以真实的冰壶场地为依准，适当改变比赛规则，便宜玩家上手，并增加的游戏的趣味性。作为一款基于OpenGL开发的游戏，该游戏实现了以下的基本图形学功能：

* 基本体素的建模表达。冰壶背景中的赛道、赛场、观众席均为立方体和多面棱台进行建模绘制；
* OBJ格式的三维网格导入。冰壶个体的顶点坐标和网格面参数为从OBJ文件中导入；
* 材质纹理的显示与编辑。背景中的布置（包括赛道、赛场、观众席）都是通过纹理贴图实现的，同时在游戏过程中，两个队伍各自的冰壶有着不同的纹理，会随着比赛进行自动切换。
* 基本几何变换能力。为了保证场景和冰壶的适应性，两者的绘制都是在基本体素经由几何变换实现绘制的。
* 基本光照模型及其编辑。场景中分别设置了可移动的光源和在冰壶赛道终点上方固定的聚光灯，在调节光照位置或冰壶运动到聚光灯下方时会产生明显效果。
* 相机漫游功能。为了在游戏过程中更好地观测冰壶的运动和碰撞状态，在整个游戏流程中设置了自适应的相机漫游并进行了平滑处理，不需要人为操作可以自动跟着冰壶进行漫游。
* 动画连续播放与截屏功能。
* 基本的碰撞检测与碰撞计算。程序能够实时监测冰壶与冰壶的碰撞、冰壶与墙壁的碰撞，并依据物理定律计算出碰撞结果，然后更新状态。
* 设计了完备的游戏规则，具有较强的可玩性。
  1. **游戏规则**

游戏一共分为四轮，每轮两个玩家轮流操作一个冰壶，通过W和S键调节初始速度的大小，通过A和D键调整初始速度方向；按IKJL键可以调节可移动光源的位置；在冰壶滑动过程中可以通过按F键来模拟摩擦，切换摩擦力状态；按C键可以对当前屏幕进行截图。最终目标是使自己的四个冰壶尽可能靠近目标圆心，根据与目标圆心的距离从小到大进行排序，最近的冰壶获得8分，其次7分，一次类推。注意：滑出或者被撞出冰壶赛道的冰壶不参与计分。

1. **模块与接口设计**

程序主要由三个模块组成，分别是Drawer模块、CurlingManager模块以及Timer模块。

* 1. **模块功能说明**

Drawer模块负责图形学变量的初始化（例如纹理参数）以及场景和冰壶的绘制；CurlingManager模块负责对程序中的所有冰壶模块进行管理，包括碰撞检测、冰壶状态更新、以及分数计算等；Timer模块则是一个简单的计时器，主要用于离散状态时间的计量。三个模块分别进行设计为类，在程序运行过程中，这三个类都分别存在一个全局对象相互进行合作调用。

* 1. **函数接口**

下面简单介绍各个模块的接口函数，Drawer模块和CurlingManager模块的具体成员变量和内部函数将在后面进一步详细说明。

* + 1. Drawer

Drawer中负责实现大部分与图形学操作有关的内容（另外少部分例如View Transformation和Light在main函数中进行设定）。其主要的对外接口函数如下：

|  |  |  |
| --- | --- | --- |
| 函数名 | 接口模块 | 功能 |
| tex\_init() | main | 纹理初始化 |
| Draw\_Background() | main | 绘制除冰壶以外的所有场景元素 |
| Draw\_Force\_Strip(double force); | main | 绘制冰壶发射初始力度蓄力条 |
| Draw\_Direction\_Line(double direc); | main | 绘制冰壶发射初始速度方向 |
| Display(double s1, double s2, double fraction); | main | 显示游戏当前状态下各个队伍的分数，以及冰壶运动时的摩擦力大小 |
| Draw\_Curling(curling\* c); | CurlingManager | 绘制出一个给定的冰壶 |

* + 1. CurlingManager

CurlingManager负责对冰壶进行管理，更新，以及计算分数等操作，主要接口函数如下：

|  |  |  |
| --- | --- | --- |
| 函数名 | 接口模块 | 功能 |
| addCurling(double x, double y, bool team) | main | 在指定位置新添加一只属于某一队伍的冰壶 |
| SetInitSpeed(int index, double theta, double Force); | main | 为一个指定的冰壶设置初速度大小和方向 |
| Calculate\_Scores() | main | 计算当前两个队伍的分数 |
| show\_all\_curlings(); | main | 显示CurlingManager中的所有冰壶，调用Drawer模块的Draw\_Curling()函数 |
| update(double time, double fraction) | main | 以当前状态为参考，在给定摩擦力情况下，更新一段时间后所有冰壶的状态。 |

* + 1. Timer

Timer模块负责对时间进行记录，主要是返回程序运行中两个节点之间的时间差，用来进行冰壶物理状态的更新；同时获取一个时间标签，用来作为瞬时截屏的文件名。主要有如下四个接口函数：

|  |  |  |
| --- | --- | --- |
| 函数名 | 接口模块 | 功能 |
| Start() | main | 获得当前时刻 |
| End() | main | 获得当前时刻并计算与开始时刻的时间差 |
| Reset() | main | 将计时器重置 |
| getTime() | main | 获得当前时刻的时间戳，作为保存截屏的文件吗 |

1. **程序流程设计**

程序运行流程大致如下图所示。在进入游戏后，首先进行相关的图形学函数和变量的加载，然后分四轮进行游戏。在每一轮中，两个队伍轮流抛出冰壶。这一过程分为几个截断。首先冰壶在起始位置初始化，然后由用户通过键盘调解初速度和方向，调节完毕后冰壶开始运动，进入第二个阶段；在运动阶段，用户可以通过F键调整模拟刷子来调节冰壶运动过程中所受的摩擦力，从而影响其加速度，在运动阶段中，每一帧会由Curling Manager进行冰壶之间的碰撞检测和状态更新，当所有冰壶都静止时进入第三阶段；第三阶段会逐渐移动相机，俯视至冰壶轨道终点，确认目标点附近的冰壶，然后更新分数，更新队伍，从头回到第一阶段。而当所有八个冰壶都完成比赛时，比赛结束，给出获胜队伍。



冰壶运动，调节加速度

初始化冰壶

调整初始速度和方向

碰撞检测

管理员系统

登入游戏

初始化冰壶

冰壶滑出

换队

更新分数

冰壶滑出

Y

N

验证运动是否停止

更新所有冰壶状态

冰壶滑出

在具体的程序设计过程中，使用了枚举变量STATE来表示游戏的进行阶段：

enum STATE{

newturn,

ready,

moving,

finish,

movingtofinish,

allfinish};

在redraw()函数中，采用switch结构对不同的状态进行分类操作以及状态更新，具体代码可见main.cpp。

main.cpp中还定义了如下的子功能函数用于一些功能的实现：

|  |  |
| --- | --- |
| 函数名 | 功能 |
| init\_Curling\_List() | 初始化冰壶显示列表 |
| readPixels() | 对当前窗口内的图像进行截屏并保存 |
| view\_ready() | 将相机转到准备阶段的视角 |
| view\_moving(int index) | 将相机设置为跟着某一个冰壶运动 |
| view\_movingtofinish() | 将相机转到赛道终点位置俯视 |
| smooth() | 在相机视角变化过程中进行平滑处理，被上述三个函数调用 |

1. **核心类说明**

下面将对核心类的成员变量以及内部函数设计进行详细说明。

* 1. **Curling类**

Curling作为作为整个程序的基础类，是最核心且最关键的，其主要用于对一个冰壶进行描述建模。其主要的成员变量如下所示：

//Location of this curling

double x;

double z;

//Mass of the curling

double mass;

//Radius of the curling

double radius;

//Speed of the curling

double vx;

double vz;

//Team of the curling

bool team;

//If the curling is stopped

bool stop;

对于一个冰壶而言，其主要的物体特征是物理位置和速度，由于冰壶始终贴着冰壶赛道，因此其相互之间有差异的分量只有x分量和y分量；其次，由于中间存在碰撞过程，因此需要为其设置质量和半径（在游戏中，所有冰壶的质量被设定为1，半径为其实际尺寸放缩后计算得）。除此之外，还需要一个变量记录冰壶所属的队伍，以便于后面CurlingManager进行计分和Drawer进行绘制。最后的stop变量用来表示冰壶是否停止，这一状态对于游戏的进行十分重要。

Curling类还具有一系列用于访问的set(),get()接口函数，这里不再赘述。

* 1. **Drawer类**

Drawer类是整个图形绘制的控制模块，其没有成员变量，但内置的成员函数较多。前面已经简单提到了它的一些对外接口函数，为了更方面的实现这些函数，其内部还有一些别的功能函数，如下所示：

|  |  |
| --- | --- |
| 函数名 | 功能 |
| LoadBitmapFile(…) | 从bmp文件中读入图片数据，被texload()调用 |
| texload(int i, char \*filename) | 加载纹理，被tex\_init()调用 |
| void drawString(char\* str) | 在显示窗口中绘制文字，被Display()调用 |
| Tex\_Cube(GLfloat n) | 绘制立方体，被Draw\_Background()调用绘制背景 |
| Draw\_Border(float x) | 被Draw\_Background()调用绘制背景 |
| Draw\_Sheet(float x) | 被Draw\_Background()调用绘制背景 |
| Draw\_Carpet(float x, float z) | 被Draw\_Background()调用绘制背景 |
| Draw\_Plank(float l, float w) | 被Draw\_Background()调用绘制背景 |
| Draw\_Audience(float l, float w) | 被Draw\_Background()调用绘制背景 |

* 1. **CurlingManager类**

CurlingManager为整个冰壶的管理模块，其主要成员变量有如下几项：

vector<curling\*> CurlingSet;

int score1;

int score0;

bool state; //which team is playing.

CurlingSet用来保存所有冰壶的指针信息，score0和score1分别表示两个队伍的得分，state用来记录目前是哪个队的冰壶在进行游戏。

CurlingManager还有如下的内部函数：

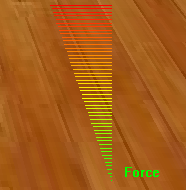
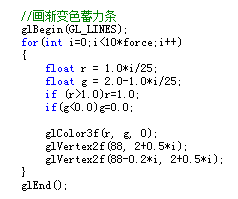
|  |  |
| --- | --- |
| 函数名 | 功能 |
| check\_collision(curling c1, curling c2) | 检测两个冰壶是否碰撞以及碰撞类型 |
| collide(curling\* c1, curling\* c2) | 两个冰壶进行碰撞 |
| collide(curling\* c1) | 冰壶与赛道和场地边缘进行碰撞 |
| teamchange() | 更改目前进行冰壶操作的队伍 |

1. **场景、纹理、光照设计**
   1. **场景设置**

****

场景的绘制是Drawer.cpp的主要功能。整个场景由地板、赛道、围板、观众席几部分组成，通过透视投影成像在屏幕上。其中地板、观众席、围板由平面（quad）表示，赛道和赛道边栏用立方体（六个平面构成）表示。对各个对象首先设置其材质系数，然后贴上纹理，以达到真实的观感。

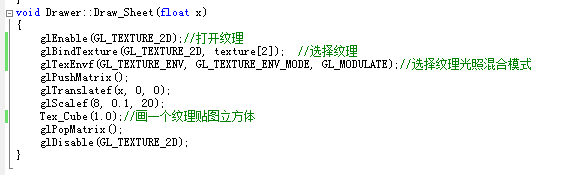
此外，通过取消深度测试之后的正投影在屏幕上显示当前比分等文字比赛信息，并在右下角用颜色渐变蓄力条显示掷壶力度大小。其中蓄力条的实现是将当前Force大小以0.1的间隔等分，然后从i=0.1开始，循环绘制水平线条，每条线的左端坐标、竖直高度以及颜色的RGB分量都是i的函数。从而实现了长度递增、颜色由绿到红渐变的效果。

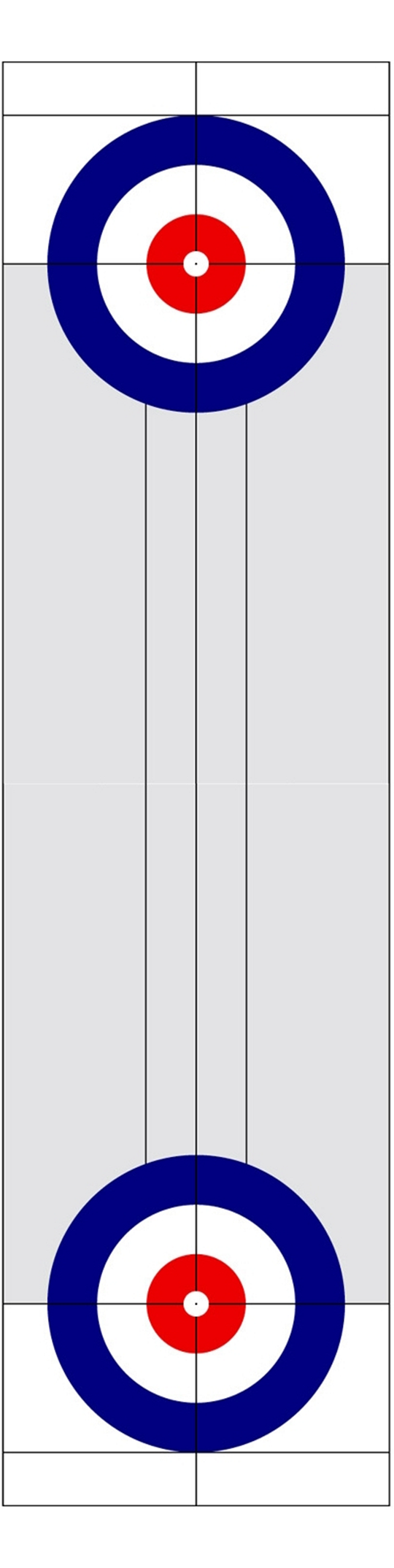
* 1. **纹理设置**

首先在void tex\_init()中导入所需的所有.bmp纹理图片，存放于数组texture[]中。每次需要贴纹理时，用glEnable(GL\_TEXTURE\_2D)打开纹理模式，用glBindTexture;选择一张纹理，并用glTexEnvf设置纹理和光照混合模式。绘制每个面之前先用glNormal3f指定面的法向量，指定面的每个顶点坐标前用glTexCoord2f指定该顶点对应的纹理坐标。

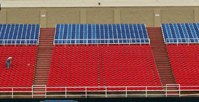
整个过程的语句较多，因此封装函数void Tex\_Cube(GLfloat n)，其功能是以当前纹理绘制一个立方体。void Draw\_Sheet(float x)是使用Tex\_Cube的一个典型示范。



我们的赛道纹理使用了冰壶标准赛道图片，此外，其他纹理贴图包括：浙江大学校牌、木质地板、观众席、赛道边缘黑色橡胶等。





* 1. **光照设置**

我们除了使用普通白色光源LIGHT0照亮整个场景，还在得分区正上方放置了朝下的聚光灯，照亮得分的圆形区域。聚光灯跟普通光源相比，需要额外设置光线方向、聚光灯发散角等参数。我们用变量表示光源的位置参数，使之可以被键盘按键控制改变，从而实现用户控制光源位置的功能。

下图是冰壶刚进入得分区时的屏幕截图，可以看到冰壶上的明显亮度分界线。



1. **其它技术细节**
   1. **碰撞检测**

冰壶的碰撞检测函数作为CurlingManager类的成员，在冰壶运动过程中起着至关重要的作用。冰壶比赛取胜的一个关键点就是冰壶的碰撞，可以碰撞己方冰壶，将其推进圈内，也可以碰撞对方冰壶，将它推出圈外。不管怎么样，冰壶与冰壶之间的碰撞都遵循着物理定律。在本游戏里，我们假设冰壶之间是弹性碰撞，根据动量守恒和能量守恒定律，从初始运动的冰壶及两个冰壶的相对位置，我们可以计算出碰撞后那一瞬间两个冰壶各自的运动状态。为了判断是否将要碰撞，可以遍历每个冰壶对象，进行两两之间的判断，如果某一帧里它们的中心距离小于等于冰壶直径，那么就进行碰撞，下一帧这两个冰壶将会有不同于原来的速度。

另外，冰壶与赛道两侧的碰撞表现为X方向速度的相反，冰壶与前方墙的碰撞表现为Z方向速度的相反，两类碰撞均伴有速度的瞬间衰减。

* 1. **自适应相机漫游**

在游戏中，相机的位置十分重要，由于冰壶的运动范围较大，因此在不同的运动过程中需要将相机调整到不同的位置，以便于玩家进行观察和操作。为了实现这一点，我们将相机的漫游设置为自适应的，即不需要用户调整，会随着游戏的不同阶段而改变。

在最初阶段，相机远离整个冰壶赛道，稍微向下俯视整个赛道；当冰壶开始运动时，相机保持与冰壶距离一定，相机中心为冰壶位置，同时相机和冰壶的连线在冰壶赛道平面上的投影与冰壶速度方向相平行；当所有冰壶都静止时，相机转化为在冰壶赛道终点的垂直俯视，用于玩家确认当前终点区的得分冰壶。

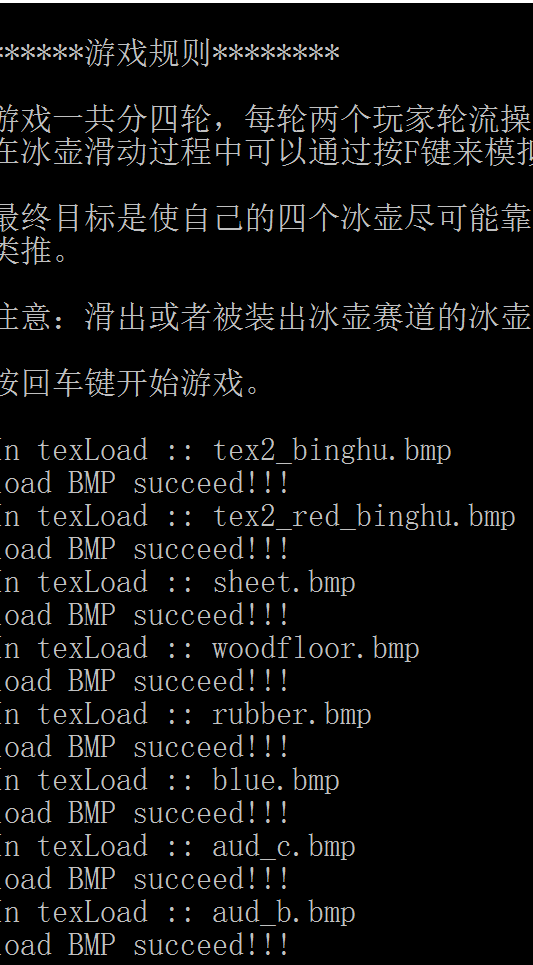
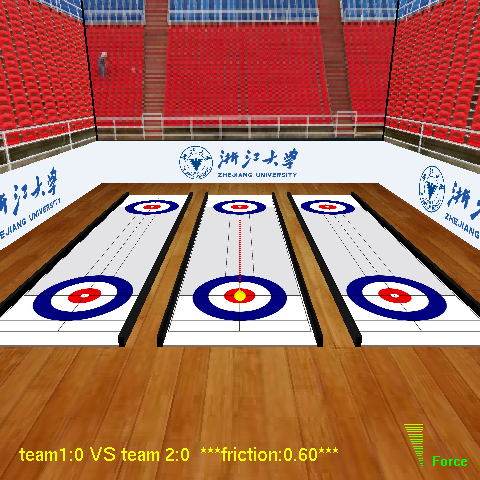
事实上，在不同阶段，只需要计算出对应的相机位置和方向通过View Transformation即可实现。但是在状态切换过程中，例如从最初的全局状态切换到冰壶开始前进的状态，此时若直接变化相机参数，会导致连续两帧图像之间没有衔接，直接从一个观测点切换到另一个较远的观测点，这样会极大的影响用户体验。出于这一考虑，我们在设计相机漫游，即相机参数变化时，采用了一个平滑算法。

具体而言，即在新的一帧绘制前，先计算两个相机参数（相机位置，View中心点位置）。将这两个点与上一帧的两个点进行进行比较，如果前后两帧的两个点都对应靠的很近，则接受新的一帧的相机参数；否则，设定一个最大阈值，以相机位置为例，若连续两帧的相机位置参数距离过大，超过了这个阈值，则将新的一帧中的相机位置设置为旧的一帧的相机位置加上阈值乘以这两点决定的方向向量。

通过以上的平滑处理，可以使得相机在漫游过程中移动地更为自然。

1. **调试结果**
   1. **初始状态**

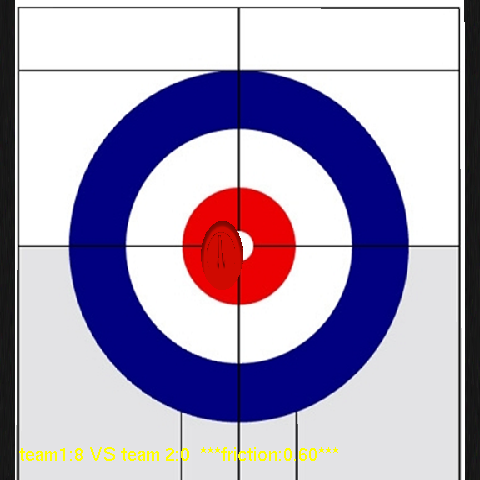
单击回车加载背景以及冰壶的材质进入游戏界面，初始比分为0：0，初始摩擦力为0.6N. 按住W或S改变初速度，右下角力度条随之改变。按住A或D改变初始方向，红色方向线随之向左或向右转动。此时按动F可以改变摩擦力的值，用来模拟冰壶运动中的擦地来减小地面的摩擦力。

 ****

* 1. **运动状态**

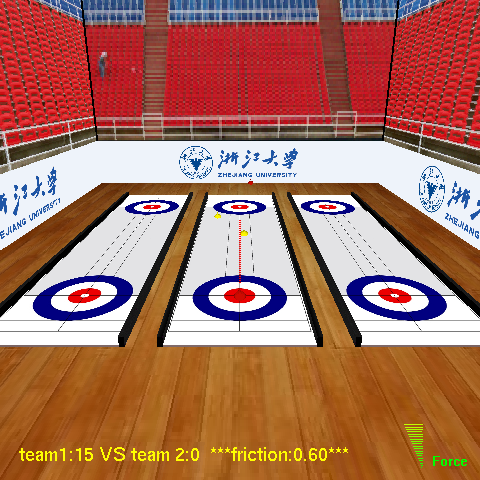
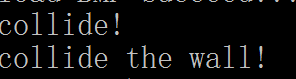
调整好合适的初速度和方向后，单机空格可以使冰壶出发冰壶由既定的初速度和方向做匀减速运动，同时相机以一定的速度跟随冰壶运动。运动中单机F键改变了地面的摩擦系数，冰壶的减速略微变慢。最终禁止后冰壶仍在赛道内，此时黄色方一队得到八分。第一掷后调整好方向和初速度后红色方二队出发，最终在红圈内停下。静止后，相机先在终点上方短暂停留后回到初始状态。显然此时二队跟接近圆圈，比分为7：8.

* 1. **碰撞**

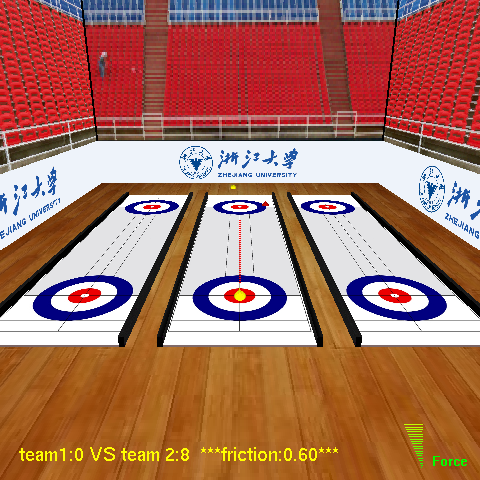
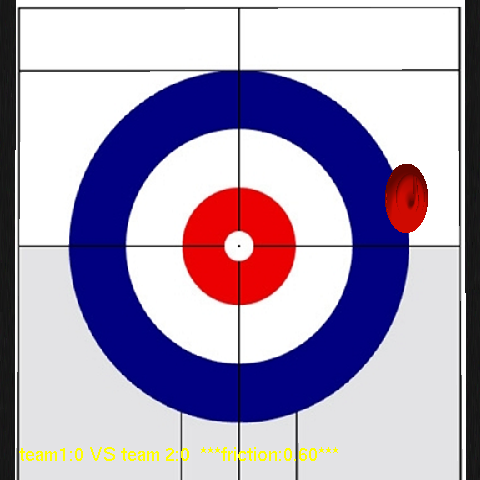
可见二队更接近红圈取得暂时领先，此时一队获得投球机会应当尽量将红球撞离红心。调整方向和力度，冰壶出发后不断目测速度与相对位置并调整摩擦系数，成功的撞击到了红色冰壶，两个冰壶发生弹性碰撞，并交换了碰撞方向上的速度，红球被撞出赛道，还与墙壁发生了碰撞，控制台输出了两次碰撞如图。最终红色方没有冰壶还在赛道上为零分，而黄色方两个冰壶都在场上各得8分和7分，比分为15：0.

* 1. **出界**

出界只判断最终的状态，如下图，前两张图为最终在界外的情况，后两张图为出界后回到赛道的情况。前者不参与积分，而后者最终回到了赛道内，被判定为有效壶，最终比分为0：8.而对于第二种情况控制台已经捕捉到了其出界后与墙壁的碰撞。

****

****

**7.5测试小结**

本游戏较好的模拟了冰壶比赛的各个环节，包括发球、擦地、碰撞、得分等情况。相机的移动可以让人很好的体会冰壶的速度和相对距离，从而合理的调整摩擦系数，有较好的交互性。