浙江大学

程 序 设 计 专 题

大 程 序 报 告

大程名称：台球游戏

小组成员：

1. **姓名 ：_____ 学号：_____ 电话：_____**

2. **姓名 ：_____ 学号：_____ 电话：_____**

3. **姓名 ：_____ 学号：_____ 电话：_____**

指导老师：许端清

2020~2021 春夏学期 2021 年 6 月 4 日

目 录

台球大程序设计项目

1 大程序简介

1.1 选题背景及意义

在本课程所提供的三个可选题目中，我们选择了较为困难的简易台球游戏。台球作为广为流传的体育运动项目，以其多变的规则、丰富的不确定性著称。也正因如此，关于台球游戏大程序的撰写相对来说较为困难。

此次选题旨在在通过团队合作提高组内成员编码能力的同时，完成游戏程序设计，实现模块化设计，并进一步巩固图形设计专题相关知识，提高编码成就感，为今后的学习和工作打下坚实基础。

1.2 目标要求

基于 libgraphics 图形库，制作一个简易的台球程序，并且支持游戏过程的读取和保存功能，即同时支持录像文件的保存和读取。对游戏内容和具体游戏方式来说，要求支持二人对战模式，并且最终能实现三种不同的游戏风格和游戏规则。

1.3 术语说明

母球：运动员从始至终用球杆直接击打的球，并利用该球运动的力量撞击其它球而得分，这个球就叫“母球”。

开球：比赛时，每局开始的第一次击球。通常由开球选手将主球放在开球区规定位置，用球杆去撞击主球，使主球去碰撞前半台的多个台球组成的球组。

开球区：必须从中开球的区域。球必须在标记范围内开出，标记后的长度不得超过两球棒的距离。

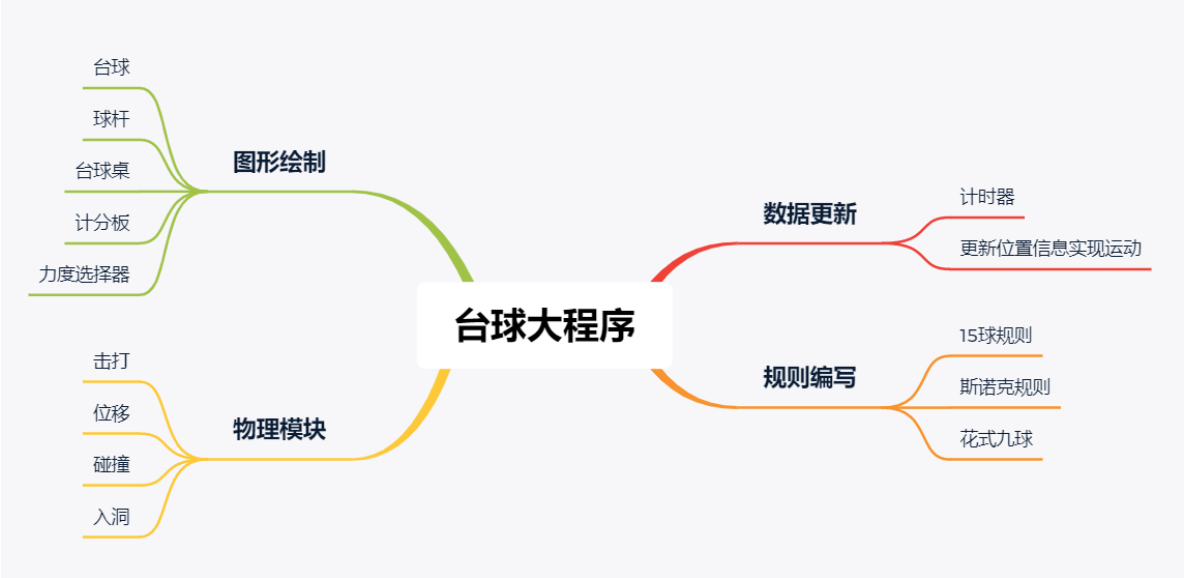
开球线：台长的 1/4，平行于底边的一条线。

中点圆：球桌正中心的圆。

置球点：开球区对面两顶袋与两腰袋对角线的交点，为置球点。

2 需求分析

2.1 功能需求

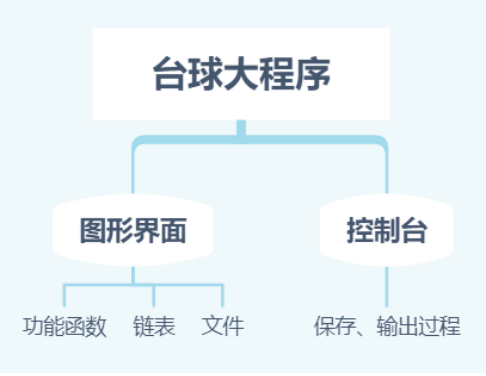


2.2 数据需求

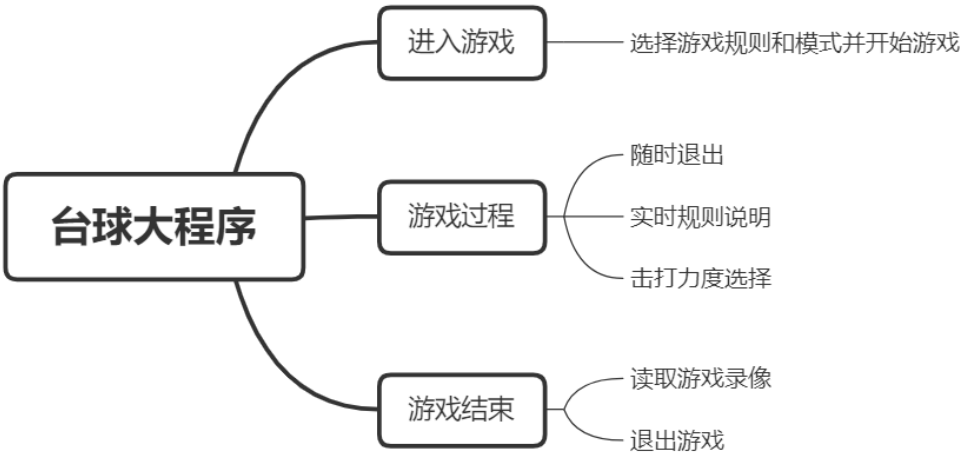
无数据需求

3 程序开发设计

3.1 总体架构设计



3.2 功能模块设计



3.3 数据结构设计

1.根据实际编码操作中的需求，同学 1 设计了如下的台球结构体。其中包含了所需台球的基本参数，在此基础上通过函数实现绘制和物理模块。

```
typedef struct Ball{
    int type;           //球的种类（花球/纯色球....)
                        //花球为Mixed 纯色球为Pure 例如 ball1 = Pure;
    int num_int;        //数字编号
    string num_str;     //字符串编号  例如 ball1.num="3";
    string color;       //颜色  例如 ball1.color="Red";
    string textcolor;   //字符串颜色
    int exit;           //是否还在场上  0：在场  1：不在场
    double x;           //x坐标
    double y;           //y坐标
    double vx;          //x速度
    double vy;          //y速度
}Ball;
```

台球 struct Ball	int type	球的种类：花色球、纯色球
	int num_int	球的数字编号
	string num_str	字符串编号，即球上画出的编号
	string color	球的颜色，即球体本身的颜色
	string textcolor	字符串颜色，即球上画出的字符串编号的颜色
	int exit	用来判断球是否还在场上，0 为仍在场上，1 为已入洞。
	double x	台球的横坐标。
	double y	台球的纵坐标。
	double vx	台球沿 x 方向的速度。

	double vy	台球沿 y 方向的速度。

2.为实现向量计算和操作，同学 2 简单设计了如下结构体。

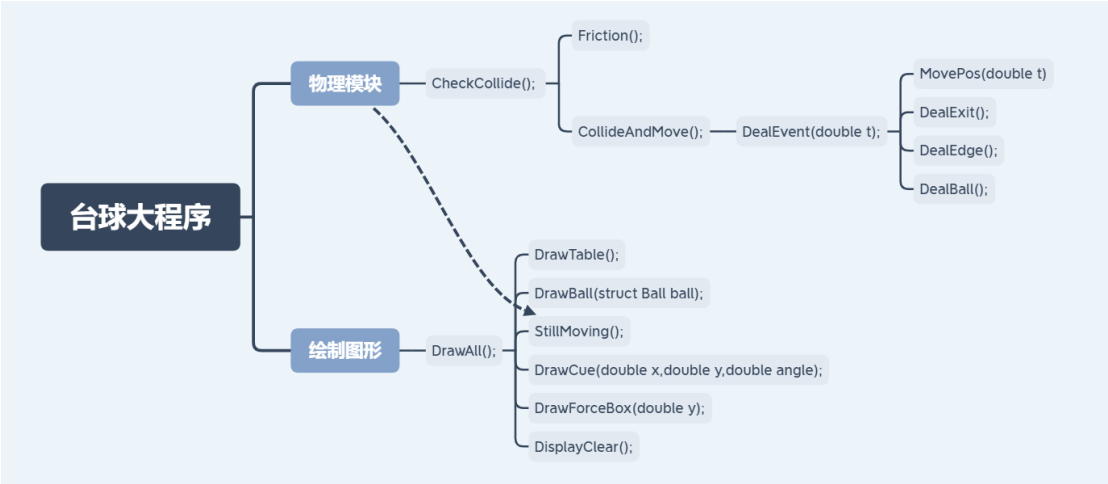
```
typedef struct vec{
    double l,d;
}vec;
```

struct vec	double l	向量长度
	double d	向量角度

3.4 源代码文件组织设计

3.5 函数设计描述

3.5.1 图形模块和物理模块



1.static void Forward(double distance)

```
/*
    * 功能：实现依照全局变量 Angle 和宏定义数据 M_PI 画一定长度的线。
    * 参数描述：distance:所画线段的长度
    * 重要局部变量定义：无
    */
```

2. static void Turn(double angle)

```
/*
    * 功能：实现将画笔朝向的角度旋转。
    * 参数描述：angle：目标旋转的角度
    * 重要局部变量定义：无
    */
```

3. static void Move(double distance)

```
/*
    * 功能：在画笔现位置上，实现依照全局变量 Angle 和宏定义数据 M_PI 移动一定距离，但不画出线。
    * 参数描述：distance:所要移动的距离。
    * 重要局部变量定义：
    * x：画笔现位置的 x 坐标。
    * y：画笔现位置的 y 坐标。
    */
```

4. void DrawBall(struct Ball ball)

```
/*
```

```
    * 功能: 按照结构体中存储的相关数据绘制台球
    * 参数描述：struct Ball ball:自定义的结构体，详见上文
    * 重要局部变量定义：
    * double FilledRadius:填充球体颜色时，需要上色的圆的半径
    * /
```

5. void DrawTable()

```
/*
    * 功能：
    * 使用该函数可以直接画出台球桌面，包括桌面框架和球洞
    * 重要局部变量定义：
    * double lx:lx= TableCenterX.
    * double ly: ly=TableCenterY.
    */
```

6. void DrawCue(double x,double y,double angle)

```
/*
    * 功能：
    * 使用该函数可以按照指定角度画出台球杆。
    * 参数说明：
    * double x:球杆端点的横坐标
    * double y:球杆端点的纵坐标
    * double angle:球杆此刻的角度
    */
```

7.void DrawBar(double a,double b)

```
/*
    * 功能：
    * 画出力度刻度
    * 参数说明：
    * double a:每个刻度之间的距离
    * double b:刻度线的长度
    */
```

8.void DrawForceBox(double y)

```
/*
    * 功能：
    * 绘制力度刻度表及文字说明，并每五格绘制红线。
    * 参数说明：
    * double y:力度选择线所在位置的纵坐标
    */
```

9.double CoorDistance(double x1,double y1,double x2,double y2)

```
/*
```

```
    * 功能：
    * 根据两点坐标计算两点间距离
    * 参数说明：
    * double x1,double y1:第一个点的横纵坐标。
    * double x2,double y2:第二个点的横纵坐标。
    * 算法说明：
    * 按照平面直角坐标系的两点间距离公式计算
    */
```

10.static void Friction()

```
/*
    * 功能：
    * 实现对实际台球游戏中摩擦的模拟。
    * 参数说明：无
    * 重要局部变量定义：
    * double v:x 方向速度和 y 方向速度矢量加和所得的实际速度。
    * double vn:经过计时器所记时间后，实际速度的变化值。
    * 算法说明：
    * 沿坐标轴方向地速度变化量与原速度的比值相同。
    */
```

11.static int Exit(double t)

```
/*
    * 功能：
    * 判断球是否入洞
    * 参数说明：
    * double t:时间间隔
    * 重要局部变量定义：
    *   double lx=TableCenterX;
    *   double ly=TableCenterY;
    * 返回值说明：
    * 球已进洞，返回 1
    * 球未进洞，返回 0
    * 算法说明：
    * 判断球洞中心和台球球体中心的距离是否小于球洞半径
    */
```

12.static int EdgeCollide(double t)

```
/*
    * 功能：
    * 实现球体和台球桌边缘碰撞
    * 参数说明：
    * double t:时间间隔
    * 返回值说明：
```

```
    * 球与台球桌边缘，返回 1
    * 球未与台球桌边缘碰撞，返回 0
    * 算法说明：
    * 判断台球结构体的横纵坐标与台球桌参数之间的关系。
    */
```

13. static int BallCollide(double t)

```
/*
    * 功能：
    * 实现台球之间的相互碰撞
    * 参数说明：
    * double t:时间间隔
    * 返回值说明：
    * 有台球发生碰撞，返回 1
    * 没有台球发生碰撞，返回 0
    * 算法说明：
    * 计算任意两个台球之间的距离。
    */
```

14. static int HasEvent(double t)

```
/*
    * 功能：
    * 判断在一段时间间隔内是否产生了新事件（入洞、球与台球桌边缘碰撞、
    台球之间相互碰撞）
    * 参数说明：
    * double t:时间间隔
    * 返回值说明：
    * 如果三个事件中任何一个都没有发生，返回 0
    * 否则，返回 1
    * 算法说明：
    * 运用以上三个函数。
    */
```

15.static void MovePos(double t)

```
/*
    * 功能：
    * 实现台球运动
    * 参数说明：
    * double t:时间间隔
    * 算法说明：
    * 认为在极短时间内做匀速运动，按照匀速直线运动公式计算新坐标。
    */
```

16.static double NextEvent(double lat)

```
/*
    * 功能：
    * 判断下一事件发生的时刻
    * 算法说明：二分找到第一个时间发生的时刻，逐次处理。
    * 参数说明：
    * double lat:时间间隔
    */
```

17.static void DealExit()

```
/*
    * 功能：
    * 实现台球入洞
    * 重要局部变量定义：
    * double lx=TableCenterX;
    * double ly=TableCenterY;
    * 算法说明：
    * 通过距离判断是否入洞
    * 若入洞修改台球结构体的 exit，并将速度设为 0。
    */
```

18.static void DealEdge()

```
/*
    * 功能：
    * 实现台球与台球桌边缘碰撞
    * 算法说明：
    * 判断台球坐标与台球桌具体参数
    * 若与台球桌壁碰撞则沿碰撞方向速度反向，沿另一条坐标轴的速度不变
    */
```

19.static void DealBall()

```
/*
    * 功能：
    * 实现台球之间的碰撞
    * 重要局部变量：
    * double Dir：用于存储两球碰撞时球心相对的角度
    * double InDs：用于存储台球半径和两球距离的差值
    * 算法说明：
    * 根据角度和距离差值，运用物理公式计算出碰撞后的小球速度参数
    */
```

20.static void DealEvent(double t)

```
/*
    * 功能：
    * 对球桌上发生的事件进行处理
```

```
    * 重要局部变量:
    * 算法说明:
    * 运用以下函数
    * MovePos(t);
    * DealExit();
    * DealEdge();
    * DealBall();
    */
```

21.static void CollideAndMove()

```
/*
    * 功能:
    * 改变调用函数时间间隔，确保处理每一个事件
    * 算法说明:
    * 利用 NextEvent(double t)函数调整时间间隔
    */
```

22.void CheckCollide()

```
/*
    * 功能:
    * 按计时间隔，触发球数据刷新
    * 算法说明:
    * 运用以下函数
    * Friction();
    * CollideAndMove();
    */
```

23.void SetSpeed(Ball* bptr,double spd,double radian)

```
/*
    * 功能:
    * 设置速度（击球等）
    * 参数说明:
    * Ball* bptr: 所要设定的台球结构体的地址
    * double spd: 设定的初速度
    * double radian: 速度的方向
    * 算法说明:
    * 将速度沿坐标轴方向分解
    */
```

24.int StillMoving()

```
/*
    * 功能:
    * 判断球是否还在运动
```

```
    * 算法说明：
    * 逐个判断场上台球速度是否为零
    */
```

```
25.void SetBalls(Ball* bptr,int size)
```

```
/*
    * 功能：
    * 用存储球数据的数组及其大小初始化
    */
```

```
26.void SetTime(double t)
```

```
/*
    * 功能：
    * 初始化计时间隔
    */
```

```
27.double VectorX(vec a)
```

```
/*
    * 功能：
    * 获取向量 X 轴上的投影
    */
```

```
28.double VectorY(vec a)
```

```
/*
    * 功能：
    * 获取向量 Y 轴上的投影
    */
```

```
29.vec NewVector(double length,double direction)
```

```
/*
    * 功能：
    * 以长度和方向新建向量
    */
```

```
30.vec ToVector(double x,double y)
```

```
/*
    * 功能：
    * 以 X 轴投影和 Y 轴投影新建向量
    */
```

```
31.vec HorResolve(vec a,double direction)
```

```
/*
    * 功能：
    * 向量在 dir 方向上的投影向量
```

```
    */
```

32. vec VerResolve(vec a,double direction)

```
/*
    * 功能：
    * 向量在 dir 方向上的垂直向量分解
    */
```

33. void DrawUserBox(int mode, Ball* Balls, int PlayerNum)

```
/*
    * 功能：
    * 在十五球规则中画出用户界面
    * 参数说明：
    * mode：用户界面的模式，根据球局开放或封闭自适应
    * Balls：规则中的球结构数组
    * PlayerNum：当前击球的玩家编号
    */
```

34. void DrawUserBall(double x,double y,int num,Ball *Balls)/*

```
/*
    * 功能：
    * 在 DrawUserBox 函数中被调用，画出在用户界面的球
    * 参数说明：
    * x：待画球的球心 x 坐标
    * y：待画球的球心 y 坐标
    * num：待画球的编号
    * Balls：规则中的球结构数组
*/
```

35. void DrawMsgBox1(); void DrawMsgBox2(); void DrawMsgBox3();

```
/*
    * 功能：
    * 画出信息框
*/
```

36. void DrawMsg1(char*,char *,char*,char*); void DrawMsg2(char*,char

,char); void DrawMsg3(char*,char *,char*);

```
/*
    * 功能：
    * 在信息框中画出信息
    * 参数说明： char*均为待输出在屏幕上的信息
*/
```

3.5.2 十五球规则模块

1.void StartMode1()

```
/*
    * 功能：
    * 对十五球规则下的游戏进行初始化
    * 即设置球数据、录像准备、设置鼠标及计时器事件等
*/
```

2.static void DrawAll()

```
/*
    * 功能：
    * 画出当前帧界面上的所有图像
*/
```

3.static void PrintStatus()

```
/*
    * 功能：
    * 给出信息：当前得分、当前行动玩家、当前违规情况和游戏结束。
*/
```

4.static void MouseEvent()

```
/*
    * 功能：
    * 处理鼠标事件：即击球和设置白球等。
*/
```

5.static void TimerEvent()

```
/*
    * 功能：
    * 处理计时器事件：即球数据刷新、每次击球后的状态判断等。
    * 算法说明：
    * 利用 CheckCollide()等函数并加静止判断。
*/
```

6. static int CheckExit()

```
/*
    * 功能：
    * 判断当前是否有球入袋，没有返回-1，有球入袋返回入袋球编号
    * 并在 MouseEvent 函数中把入袋球加入到入球链表中
*/
```

7、static int CheckFirstCollision()

```
/*
    * 功能：
    * 判断当前杆击打到的第一个球是否符合规则
    * 没有击打到球返回 0，击打到球返回被击打球编号
*/
```

8、static int GameOver()

```
/*
    * 功能：
    * 判断当前游戏是否正常结束
    * 正常结束返回 1，非正常结束返回 0
*/
```

9、static void EndGame()

```
/*
    * 功能：
    * 结束游戏，输出胜利信息
*/
```

10、static void DealBreakRule()

```
/*
    * 功能：
    * 处理犯规操作，输出犯规信息
*/
```

11、static void PrintStatus()

```
/*
    * 功能：
    * 输出当前球局状态
*/
```

12、

```
typedef struct Player{
    int num;
    int color;
}Player;
static Player player[2];
/*
    * “玩家”结构体
    * 结构体成员说明
    * num:玩家编号
    * color:玩家所属球的型号
*/
```

```
13、typedef struct ExitBall{
    int num;
    int cue;
    struct ExitBall *next;
}ExitBall;
/*
    * “入袋球”结构体
    * 结构体成员说明
    * num:球编号
    * cue:该球入袋时杆的编号
    * next：下一个入袋球结构体的指针
*/
```

3.5.3 斯诺克规则模块

1.void StartMode2()

```
/*
    * 功能：
    * 对斯诺克规则下的游戏进行初始化
    * 即设置球数据、录像准备、设置鼠标及计时器事件等
*/
```

2.static void DrawAll()

```
/*
    * 功能：
    * 每隔一个极小时间（20ms）清除屏幕并重新绘制桌面状态和其他按钮
*/
```

3.static void PrintStatus()

```
/*
    * 功能：
    * 给出信息：当前得分、当前行动玩家、当前违规情况和游戏结束。
*/
```

4.static void MouseEvent()

```
/*
    * 功能：
    * 处理鼠标事件：即击球和设置白球等。
*/
```

5.static void TimerEvent()

```
/*
```

```
    * 功能：
    * 处理计时器事件：即球数据刷新、每次击球后的状态判断等。
    * 算法说明：
    * 利用 CheckCollide()等函数并加静止判断。
    */
```

```
6.static void CheckNextBall()
```

```
/*
    * 功能：
    * 查找下一次击球的可用球状态，并存储在全局变量中。
*/
```

```
6.static void CheckStatus()
```

```
/*
    * 功能：
    * 每一次击球完成后，计算得分及犯规情况
    * 并调用其他函数处理彩球、白球放回。
*/
```

```
7.static int TrueFirstBall()
```

```
/*
    * 功能：
    * 确定母球第一个击中的球是否为正确目标球，并返回 1（正确）或 0。
*/
```

```
8.static int ResetColorBall()
```

```
/*
    * 功能：
    * 处理彩球放回。
    */
```

3.5.4 花式九球规则模块

```
1.void StartMode2()
```

```
/*
    * 功能：
    * 对花式九球规则下的游戏进行初始化
    * 即设置球数据、录像准备、设置鼠标及计时器事件等
*/
```

```
2.  static void StartGame()
```

```
/*
    * 功能：
```

```
    * 对台球数据进行初始化
*/
```

```
3.static void DrawAll()
```

```
/*
    * 功能：
    * 绘制所有图像
*/
```

```
4. static void Legal()
```

```
/*
    * 功能：
    * 实现换人
*/
```

```
5.static void TimerEvent(int event)
```

```
/*
    * 功能：
    * 处理计时器事件：即球数据刷新、每次击球后的状态判断等。
    * 算法说明：
    * 利用 CheckCollide()等函数并加静止判断。
*/
```

```
6. static void MouseEvent(int x, int y, int button, int event)
```

```
/*
    * 功能：
    * 处理鼠标事件。
*/
```

```
9.void GameResults()
```

```
/*
    * 功能：
    * 输出游戏结果。
*/
```

3.5.5 UI 和录像模块

```
1.void InitUI()
```

```
/*
    * 功能：
    * 显示主菜单。
*/
```

2. void DrawUI()

```
/*
    * 功能：
    * 画出主菜单的按钮，并判断是否出现点击。
*/
```

3.static void MouseEvent()

```
/*
    * 功能：
    * 获取鼠标参数以供 UI 使用
*/
```

4.static void TimerEvent()

```
/*
    * 功能：
    * 每隔 20ms 刷新主菜单。
    */
```

5. void InitSave(Ball* Balls,int BallCnt)

```
/*
    * 功能：
    * 在游戏开始时，用球数据的指针初始化录像存储的数据。
    * 打开 newrecord 文件以便写入。
*/
```

6. void SaveStep(double Speed,double CueAngle,int p)

```
/*
    * 功能：
    * 将每一步击球的速度、角度及玩家信息写入文件。
*/
```

7.void InitRead()

```
/*
    * 功能：
    * 获得录像文件名并初始化球桌数据。
*/
```

8.static void MouseEvent(int x,int y,int button,int event)

```
/*
    * 功能：
    * 获取鼠标位置，便于 UI 利用。
    */
```

1. static void TimerEvent(int event)

```
/*
    * 功能：
    * 刷新 UI 和桌面数据，给出按钮（“主菜单”和下一步）。
    * 给出玩家信息及录像结束信息。
*/
```

2. static void NextStep()

```
/*
    * 功能：
    * 读取文件，得到下一次击球的初始状态。
*/
```

3.static void GetFileName()

```
/*
    * 功能：
    * 通过控制台窗口得到用户想打开的录像文件名，存储在全局变量中。
*/
```

4 部署运行和使用说明

4.1 编译安装

1.新建项目文件。

2.项目->添加。按照上文所述源代码文件组织设计将源代码添加至项目文件。

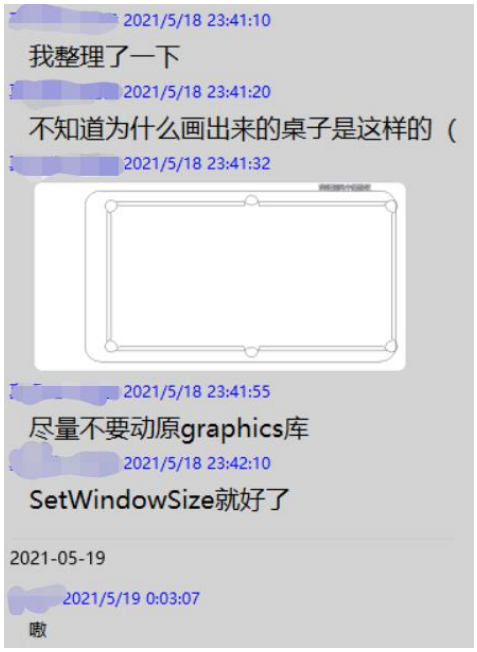
3. F12 编译全部

4. 成功获得 exe 程序

4.2 运行测试

2021.5.18

运行后发现台球桌偏移。解决方法：用 SetWindowSize 代替修改原 graphics 库。

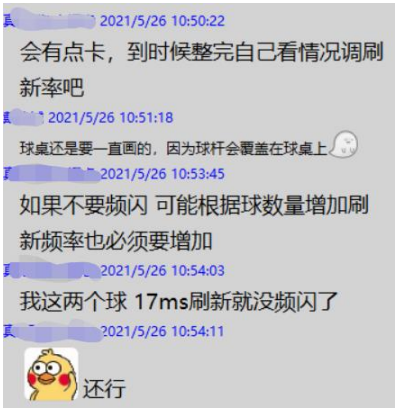


2021.5.19

解决上述问题，具体修改情况如下。此外，为使不同模块编码更加便捷，将坐标常量宏定义。

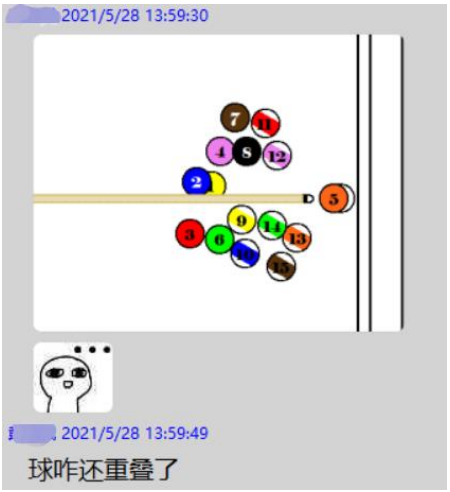
2021.5.26

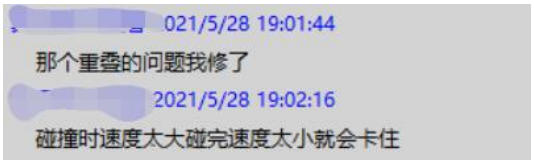
发现频闪问题。解决办法：按照实际情况提高刷新频率。



2021.5.28

发现台球重叠问题。问题出现原因：碰撞时速度太大碰完速度太小就会卡住。





2021.5.30-2021.6.2

发现台球无法进洞问题。解决办法：将 physics 里的 lx 和 ly 换成 TableCenter，并调整参数。

4.3 使用操作

1.打开台球游戏应用程序，进入初始界面。

2.在前三项中自行选择游戏规则和模式，并单击鼠标，进入游戏。

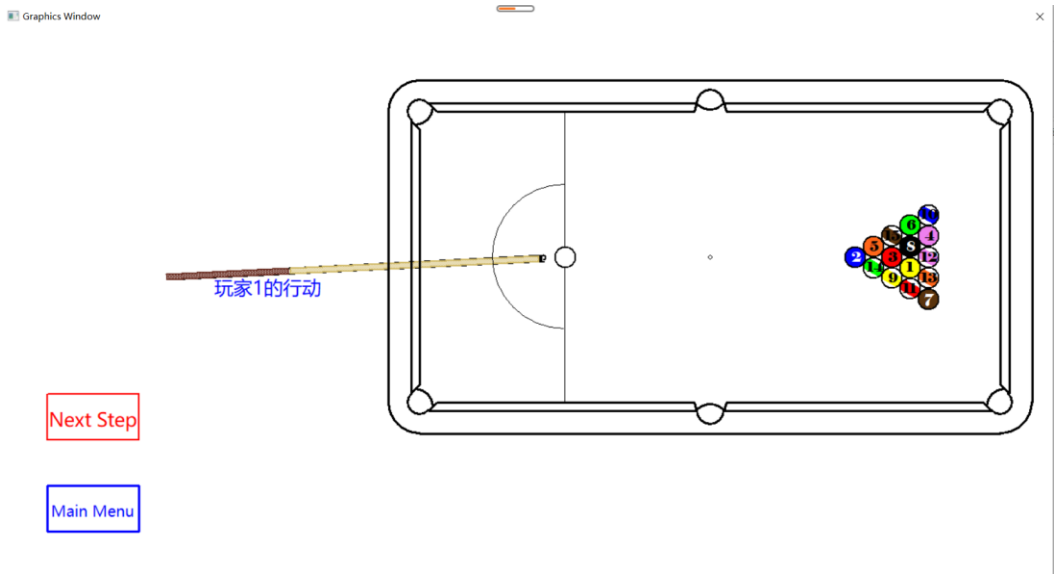
3.游戏过程中的操作

1)台球杆跟随鼠标旋转，确定方向后单击鼠标，出杆击球。

2)最左边的力度刻度选择器。绿线所在位置为此时所选速度，击球力度为单击鼠标时绿线所在位置所代表的力度。

3)实时规则说明。说明此时是玩家 1 或玩家 2 击球，并说明击球顺序或击球限制，并实时更新双方得分情况或进球情况。

4)游戏过程中可随时点击左下角退出按钮退出游戏。



4.读取录像。可按照每次击球的情况逐次观看游戏录像。

5.退出游戏。单击游戏页面或初始页面的退出按钮，退出程序。

5 团队合作

5.1 任务分工

同学 1：基础图形设计（球桌、球、球杆），绘制模块函数设计（信息框及信息打印），台球数据设计。15 球规则。

同学 2：物理模块（包含一个向量库和碰撞、判断入洞、设置速度等函数）。录像功能（录像的保存和读取）。简易主菜单 UI 设计。斯诺克规则。

同学 3：报告总结及撰写。花式九球规则。

5.2 开发计划

第 0 周：确定选题。

第 1-2 周：完成基础图形设计。

第 3 周：确定基础图形参数。

第 4-5 周：完成物理模块撰写调试。

第 6 周：完成规则编写，整合。

5.3 编码规范

严格按照本课程参考文件中的编码规范。

5.4 合作总结

确定选题

05/04-05/08

在经过简单讨论后，确定本次大程序选题——台球游戏大程序。并初步讨论了大致方向，在查阅相关资料，将大程序分为几个模块。

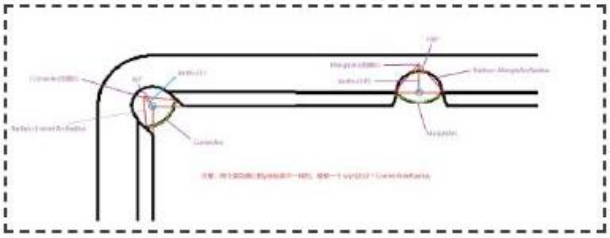
初步绘制图形

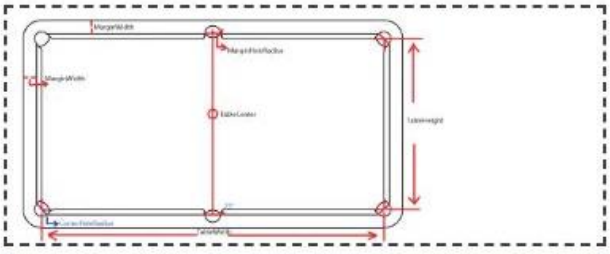
05/12

初步设定台球桌参数
完成台球绘制函数

确定台球桌参数
并确定详细数据

05/21



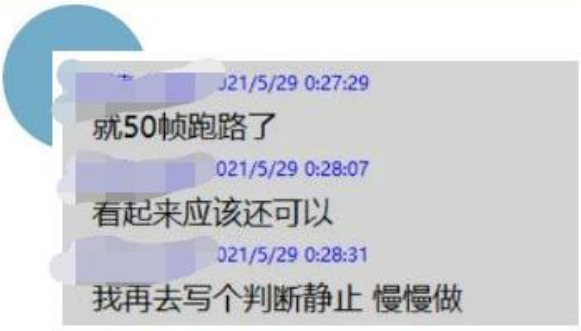


物理模块初版

05/22

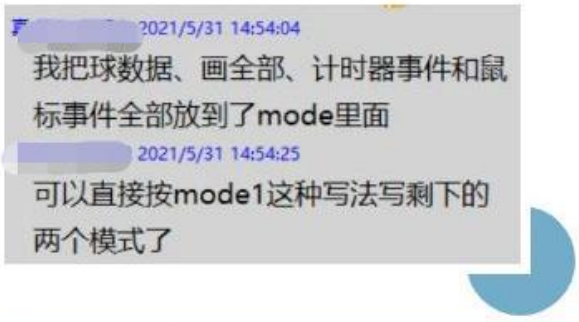
判断静止函数

5/29



各模块整合完毕

5/31



规则撰写完成
终版出炉

06/14

5.5 收获感言

同学 1：从最开始的对台球桌一笔一划的 DrawLine 到对每一个球的花色自适应绘制，再到对力度显示器的绘制，许多复杂的图形都是一条条基础指令绘制出来的。游戏规则模块虽然很难，需要将许多复杂的游戏规则转化为基础的逻辑判断，但可以设立一些状态参量来辅助判断，对于每次运行先判断再依据状态参量执行，将判断和执行部分分离开，从而帮助程序员理清思路。Bug 在所难免，debug 有时也非常痛苦，但是为程序一块块添砖加瓦的过程却是快乐享受的。

同学 2：在较复杂的程序设计过程中，自顶向下、逐步求精的思想对简化编程难度十分有帮助。在每个看起来比较复杂的 bug 修复和新功能添加中，只要能够沉下心来仔细分析问题的来源，解决问题是相对简单的。对并列模式的程序来说，分成多个子文件并且用 static 进行变量声明是相当有助于逻辑的。

同学 3：利用模块化设计思想原则，首先将问题划分为几大模块，再逐步细分为具体的待解决的问题。利用简单的物理公式、数学知识以及建议逻辑推理实现所需功能。将模块化思想运用于每一次的编程进程中，将会对以后的工作生活有所帮助。

6 参考文献资料

[1]做游戏，学编程（C 语言） 21 台球_童晶的博客-CSDN 博客