杭州电子科技大学
HANGZHOU DIANZI UNIVERSITY

2021 游戏程序设计报告

课程名称：游戏程序设计

任课老师：李仕

姓名：郑语童、费博雯、王艳、胡欣雨

学号：19220207 、19220203 、19220206、19220204

专业：数字媒体技术

简介：

1.游戏基本资料

- 游戏名称：飞鸟
- 游戏类型：棋牌类游戏

2.游戏概要

玩家通过独立作战或与对家配合，通过掷骰子数，决定自己棋子在棋盘上的行进步数，以全部棋子最先到达终点的游戏者为最后胜利者。可以锻炼玩家的策略意识以及灵活变通。

3.游戏界面分析

3.1 游戏开始界面

从初始界面切换至游戏开始界面（展示棋盘图案、玩家位置）

3.2 游戏内容界面

- 界面操作说明：

骰子：按左键进行掷骰，得出掷骰点数。

起飞：选择行动的棋子，棋子按点数行走。在掷得 6 点后，方可将一枚棋子由“基地”起飞至起飞点，并可以再掷骰子一次，确定棋子的前进步数。

棋盘：棋子按照棋盘上的格子进行运动。

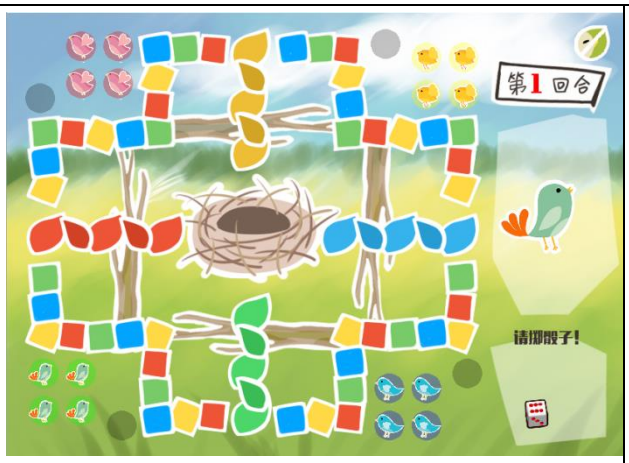
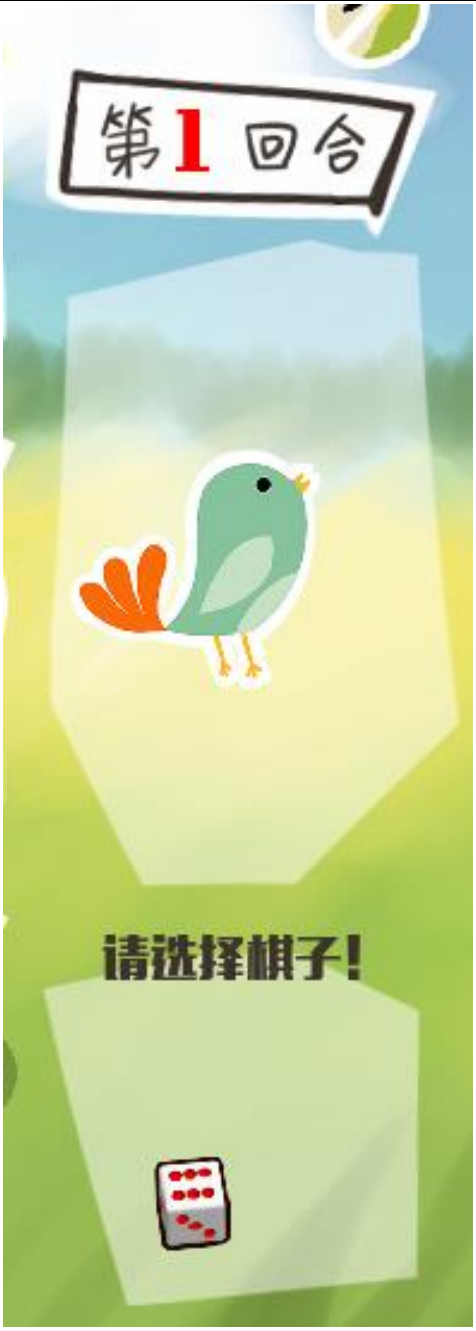
玩家：显示此次掷骰是由哪个玩家进行的。

获胜：当有玩家的所有棋子到达终点时，则该玩家获胜。

结束：当所有棋子到达终点时，则游戏结束。

- 界面说明：

界面	截图	说明界面内容解说
初始界面		初始界面

棋子飞行界面 （棋盘界面）		棋盘 MAP
掷骰界面		

某棋子到达终点界		
玩家选择棋子界面 （棋子制作完成）	 	棋子

4.音乐音效

1、背景音开启 

2、背景音关闭 

3、投骰子音效

4、棋子碰撞音效

5、棋子飞行音效

5.游戏世界各元素定义

（列表对游戏世界包含的各个对象进行说明）

对象	作用	属性
图片资源	提供显示界面的图片	棋子图、功能按键图、骰子图
棋子（鸟）	玩家操控的对象	通过玩家的指令在棋盘上移动

骰子	控制棋子的移动	棋子根据骰子的点数以及玩家的选择进行移动
棋子（蛋)	表示到达终点	当棋子（鸟）到达终点时，鸟的形态会变成蛋，回到出发点。

6.游戏逻辑实现（举例如下）

6.1 初始化游戏世界

（1）加载图片和音频资源

加载棋盘图片、棋子图片、玩家图片、骰子图片。

（2）初始化地图

棋子全部位于初始位置。

（3）初始化鼠标状态

将鼠标状态置为未按下。

6.2 游戏循环

（1）输入

鼠标左键按下：设置鼠标状态为左键按下。

鼠标左键松开：设置鼠标状态为未按下。

鼠标右键按下：设置鼠标状态为右键按下。

鼠标右键松开：设置鼠标状态为未按下。

（2）逻辑处理

1.骰子逻辑

//初始化棋子位置

//初始化棋子信息

Player_logic()：

骰子抛到 6，则棋子可以飞行。并且玩家可以再次投骰。

骰子未抛到 6，则棋子无法飞行，下一位玩家。

AutoPlayer_logic：

骰子抛到 6，则棋子可以飞行。并且玩家可以再次投骰。

骰子未抛到 6，则棋子无法飞行，下一位玩家。

```
void Game::LogicRoll()
{
    srand(time(NULL));
    if (isClickShaizi)
    {
        Roll.before_click = Roll.after_click;
        Roll.after_click = rand() % 6 + 1;
        isClickAnimation = true;
        soundShaizi.play();
        isClickShaizi = false;
        isTimetoClick = false;
        isTimetoShowShaizi = false;
        isTimetoChoose = true;
        pic = 0;
    }
}
```

2. 小鸟的行走逻辑。

当掷到六时，玩家可以选择小鸟进入准备状态或者行走。当没有掷到六时，需要根据准备区是否有小鸟来实现走或不走。没有掷到六时的部分代码逻辑的实现如下：

```
else if (Roll.after_click > 0 && Roll.after_click < 6) ////骰到不是6
{
    for (int i = 0; i < 4; i++)
    {
        if (player[round_player].chess[i].IsReady || player[round_player].chess[i].IsOut)
            ready_chess++;
    }
    if (ready_chess == 0)
    {
        isTimetoChoose = false;
        choose_chess = -1;
        round_player = (round_player + 1) % 4;
        isTimetoShowShaizi = true;
        isTimetoClick = true;
        HaveChoose = false;
        AnimationOver = false;
        if (round_player == 0)
        {
            round++;
        }
    }
    else if (ready_chess == 1)
    {
        isTimetoChoose = false;
        for (int i = 0; i < 4; i++)
        {
            if (player[round_player].chess[i].IsReady || player[round_player].chess[i].IsOut)
            {
                choose_chess = i;
                break;
            }
        }
        if (choose_chess != -1 && !HaveChoose)
```

小鸟的行走步数由 step 来记录。当小鸟的行走步数小于 50 步时，还要判断小鸟是否需要跳跃或飞行。当小鸟行走后所在的格子为同色，则可能发生跳跃或飞行。按照地图的规定，小鸟出门的第二格就是同色的，然后每隔四格就是同色的格子。当小鸟恰好走到第 14 步或第 18 步时，小鸟会实现跳跃+飞行，共跨越 16 步。具体代码逻辑的实现如下：

```
if (player[round_player].chess[choose_chess].IsReady == true && player[round_player].chess[choose_chess].IsOut == false && player[round_player].chess[choose_ch
{
    player[round_player].chess[choose_chess].old_step = player[round_player].chess[choose_chess].step;
    player[round_player].chess[choose_chess].IsReady = false;
    player[round_player].chess[choose_chess].IsOut = true;
    player[round_player].chess[choose_chess].step += Roll.after_click;
    if ((player[round_player].chess[choose_chess].step + 2) % 4 == 0 && player[round_player].chess[choose_chess].step < 50)
    {
        if (player[round_player].chess[choose_chess].step == 14)
        {
            player[round_player].chess[choose_chess].step += 16;
            Fly = true;
        }
        else if (player[round_player].chess[choose_chess].step == 18)
        {
            player[round_player].chess[choose_chess].step += 16;
            Fly = true;
        }
        else
        {
            player[round_player].chess[choose_chess].step += 4;
            Jump = true;
            Jump = true;
        }
    }
}
```

当 step 的步数刚好等于 55 步时，小鸟回到鸟巢，产出一颗鸟蛋。具体代码逻辑的实现如下：

```
if (player[round_player].chess[choose_chess].step == 55)
{
    Back = true;
    player[round_player].chess[choose_chess].step = 0;
    player[round_player].chess[choose_chess].IsReady = false;
    player[round_player].chess[choose_chess].IsOut = false;
    player[round_player].chess[choose_chess].IsWin = true;
    player[round_player].chessWin_Count++;
}
```

当 step 步数大于 55 步时，小鸟触底后要往回退。具体代码逻辑的实现如下：

```
if (player[round_player].chess[choose_chess].step > 55)
{
    player[round_player].chess[choose_chess].step = 110 - player[round_player].chess[choose_chess].step;
}
```

游戏逻辑的代码很长，但核心逻辑就是上述的不同 step 步数相对应的逻辑代码。

3.碰撞检测

小鸟行走的过程中，可能会和其他不同色的小鸟走到同一格，这时就会将那只小鸟撞回。所以，小鸟的每次行走都要通过碰撞检测 HitTest()函数检测。

HitTest()函数的思路是遍历所有非同色的小鸟，若其位置与当前小鸟的位置相同，则撞回。具体代码逻辑的实现如下：

```
void Game::HitTest(Chess Now_Chess)
{
    Vector2i NowPosition;
    NowPosition.x = Now_Chess.position_x;
    NowPosition.y = Now_Chess.position_y;

    for (int i = 0;i < 4;i++)
    {
        if (i == Now_Chess.Host)
        {
            continue;
        }

        for (int j = 0;j < 4;j++)
        {
            Vector2i TestPosition;
            TestPosition.x = player[i].chess[j].position_x;
            TestPosition.y = player[i].chess[j].position_y;
            if (NowPosition.x == TestPosition.x && NowPosition.y == TestPosition.y)
            {
                soundHit.play();
                Back = true;
                player[i].chess[j].IsOut = false;
                player[i].chess[j].old_step = player[i].chess[j].step;
                player[i].chess[j].step = 0;
            }
        }
    }
}
```

6.3 游戏结束

当有一玩家的四个棋子全部到达终点，则游戏结束。

游戏 SWOT 分析

S 优势	界面美观，画风可爱，音效解压。适合团体活动时缓解无聊。	O 机会	总会有无聊的人喜欢可可爱爱的消磨时间小游戏，就好比于之前爆火的合成大西瓜等简单小游戏。
W 劣势	只具备了传统飞行棋的功能，没有什么创新模块。	T 威胁	市面上存在许多种类的飞行棋，且其完善度与创新度与我们的相比较高

产品基本面分析：

游戏的卖点：可爱画风，团体游戏。

用户分析：当朋友聚会无聊时可以玩的消磨时间小游戏。

竞争分析：

市面上存在许多种类的飞行棋，且其完善度与创新度与我们的相比较高，所以可能会存在竞争困难的情况。

盈利分析：

通过加入广告获得盈利（如果有人愿意的话）