项目说明

版本号	修改时间	修改者	内容
0.1	2024/10/4	黄炫豪	原型制作，创建文档
0.2	2024/10/5	黄炫豪	收尾，修改文档

一、项目架构

1. **人机模式：**由玩家与电脑人机进行作战
2. **玩家模式：**在同一时空内，由两个玩家共同操作
3. **联网模式（未做）:**测试后发觉网络不稳定，故舍弃
4. **设置：**进行棋盘大小、人机难度的设置

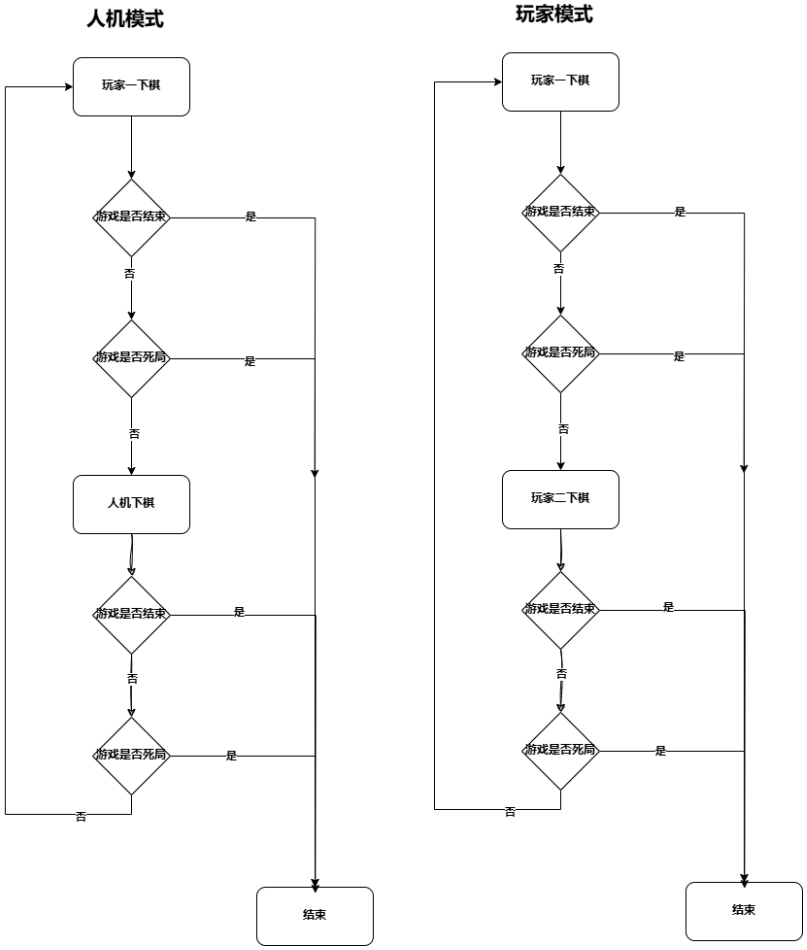
序号	棋盘大小	人机难度
1	3	简单
2	4	一般
3	5	困难

二、思路概述

1. **棋盘生成**
   - 在初始化时，从GameManager处获取棋盘大小，动态生成并放在同一父项下

```
size = GameManager.instance.grid_size;
for (int i=0;i<size;++i)
    for(int j=0;j<size;++j)
    {
        Button go = Instantiate<Button>(Btn_Clone);
        go.gameObject.SetActive(true);
        go.name = i + "_" + j;
        go.transform.parent = Btn_Parent.transform;
        go.transform.localScale = new Vector3(1, 1, 1);
        grids[i * size + j] = go;
    }
```

2. **游戏回合切换**
   - **人机模式：**玩家下棋后，模拟若干秒延迟后，人机基于简单算法下棋
     - Invoke("Paint_PVE", timer);
   - **玩家模式：**两个玩家轮流下棋，直至获胜



3. **判断胜负**
   - 依次检查每行每列及对角线是否为同一元素

   同时，要判断是否死局

```
//检查是否死局
bool Is_Lock = true; ;
for(int i=0;i<size;++i)
{
    for(int j=0;j<size;++j)
    {
        if(board[i,j]==0)
        {
            Is_Lock = false;
            break;
        }
    }
}
if(Is_Lock)
{
    return -1;
}
```

4. **交互提示**
   - 对以下行为作出提醒

序号	行为	效果
1	下棋到不是空格子	Tips提示
2	玩家获胜	Tips提示
3	死局了	Tips提示
4	当前操作玩家	选中框切换

5. 屏幕**适配**
    - 基于长宽比判断横竖屏状态，进行UI调整

横屏

竖屏

三、人机难度设计

1. **难度1：**最简单的人机，随机选取棋盘一个空格进行下棋

```
int[] pos = new int[size * size];
int none_count = 0;//空闲的格子
for (int i = 0; i < size; ++i)
{

    for (int j = 0; j < size; ++j)
    {
        if (board[i, j] == 0)
        {
            pos[none_count++] = i * size + j;


    }
}
    }
int count = (int)Random.Range(0, none_count);
board[pos[count] / size, pos[count] % size] = 2;
```

2. **难度2：**在难度1的基础上，选取能获胜的空格进行下棋，否则随机选取空位下棋

```
for (int i=0;i<size;++i)
{
    for(int j=0;j<size;++j)
    {
        if(board[i,j]==0)
        {
            board[i, j] = 2;
            if(Check_Win()==2)
            {
                grids[i* size + j % size].image.sprite = grid_state[2]
                Show_Win(2);
                return;
            }
            board[i, j] = 0;
            pos[none_count++] = i * size + j;
        }
    }
}
int count = (int)Random.Range(0, none_count);
board[pos[count] / size, pos[count] % size] = 2;
```

3. **难度3：**在难度2的基础上，先选择能获胜的下棋，再阻止玩家“将军”

```
    if (board[i, j] == 0)
    {
        board[i, j] = 2;
        if (Check_Win() == 2)
        {
            grids[i * size + j % size].image.sprite = grid_state[2];
            Show_Win(2);
            return;
        }
        board[i, j] = 1;
        if (Check_Win() == 1)
        {
            tarX = i;
            tarY = j;
        }
        board[i, j] = 0;
        pos[none_count++] = i * size + j;
    }
}
```