1. 程序描述

游戏核心，负责单局游戏运行逻辑和游戏信息处理，接收玩家操作序列，处理后，更新并输出游戏当前信息。

1. 功能
2. 储存当前游戏所有玩家和环境信息
3. 接收玩家操作
4. 处理玩家操作
5. 处理游戏固定进程
6. 更新当前游戏所有玩家和环境信息
7. 输出当前游戏所有玩家和环境信息
8. 性能

CPU i7-7700HQ，内存16G的情况下，每秒可处理单人操作5000次以上

1. 输入

Python字典，形如{'id': number, 'up': 0, 'down':0, 'left': 0, 'right': 1 }

Number 字符串要求每个字符是整数，up down left right 对应0或1

Number是玩家id，up down left right 对应0表示没有进行，1表示进行了

1. 输出

输出python字典，output\_dict = {'info':self.information,'tanks': self.tanks, 'bulls':self.bulls, 'obs': self.obs, 'props': self.props , 'safe': self.safe}

字典格式说明

dict = {

'info':[i,i,i,i,i,i] 6个int---扩展、获胜者id、坦克数、子弹数、障碍物数、道具数

'tanks':[[i,i,i,i,i,f,f] , []] 5个int2个float为一个数组，多个数组拼凑 ---坦克id、坦克生命、坦克弹药、坦克杀敌数、方向、x、y

'bulls':[[i,i,i,f,f], [],[]] 3个int两个float为一个数组，多个数组拼凑---子弹发射者id、子弹id、是否爆照、x、y

'obs':[[i,i] ,[i,i]] 坐标对

'props':[[i,i,i,i,i], []] 5个int为一个数组 ---- 道具id、类型、状态、x、y

'safe':[i,i,i,i,i,i,i,i] 8个int

}

1. 数据结构和算法
2. class GlobalInfo: 游戏的全局参数和全局变量

time 游戏时间秒数

poison = 0.01 非安全区毒性，每帧扣除坦克poison的血量

tank\_v = 0.1 坦克移动速度

ammo\_v = 0.5 子弹移动速度

ammo\_attack = 10 子弹攻击力

tank\_life = 100 坦克初始生命值

tank\_ammo = 50 坦克初始子弹数

item\_hp = 10 医疗包道具加的血量

item\_ammo = 10 子弹包道具加的子弹数

def \_\_init\_\_(self):

self.ammo\_num = 0 游戏中存在的子弹数

self.brick\_list = [] 障碍物列表，每个元素包含坐标

self.item\_list = [] 道具列表，每个元素包含坐标和道具类型0表示医疗包，1表示子弹包

self.operate\_queue = Queue() 操作队列，每个元素是字典operate\_dict

self.operate\_dict = {} 操作字典

self.ammo\_list = [] 当前游戏中所有子弹的列表

self.tank\_list = [] 当前游戏中所有坦克的列表

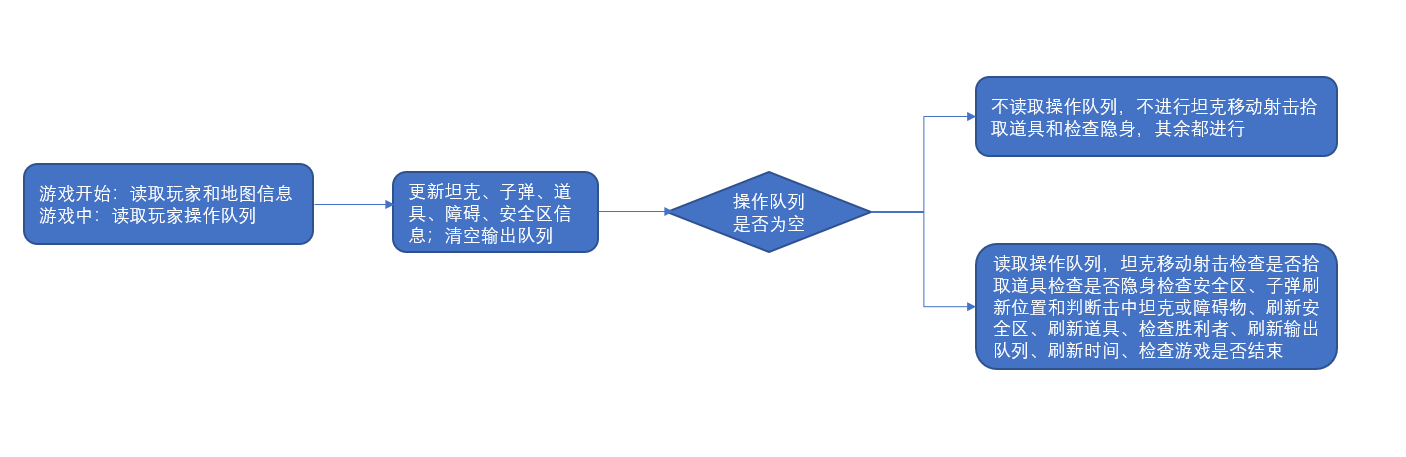
self.brick\_changed = [] 当前游戏中所有改变了的障碍物列表

self.item\_changed = []当前游戏中所有改变了的道具列表

self.winner = -1 胜利者，-1表示无，有胜利者则为胜利者的id

self.output\_dict= {'info':self.information,'tanks': self.tanks, 'bulls':self.bulls, 'obs': self.obs, 'props': self.props , 'safe': self.safe} 输出当前玩家和游戏信息

1. class Tank: 坦克
2. def shoot(self, fire): 射击方法，fire为1则生成子弹
3. def drive(self, up, down, left, right): 行驶，up表示前进down后退，left和right分别是向左向右转，取值都为0和1，1表示执行，0表示不执行
4. def is\_in\_circle(self): 判断坦克是否在安全圈，在则返回true，否则返回false
5. def \_\_init\_\_(self, tank\_id, hp, ammo, kill, direction, x, y,info):初始化坦克相关信息，info是游戏全局信息，x和y分别表示位置x和y坐标，kill表示杀敌数
6. class Ammo: 子弹
7. def move(self): 子弹移动，每帧移动一次
8. def refresh(self): 刷新全局信息，包括子弹击中坦克、子弹击中障碍物、子弹击中边界
9. def \_\_init\_\_(self, tank\_id, ammo\_id, x, y, direction, info): 初始化信息
10. class Brick: 障碍物
11. def \_\_init\_\_(self, x, y): 初始化自身位置和是否存在信息
12. def disappear(self): 消失，将是否存在置为否
13. class Grass: 草
14. def \_\_init\_\_(self, x, y): 初始化位置
15. class Item: 道具
16. def \_\_init\_\_(self, x, y, type\_id): 初始化位置和类型
17. def disappear(self): 消失，将是否存在置为否
18. class Circle: 安全圈
19. def refresh(self): 刷新安全圈，每10秒将安全圈向内缩小一个单位，直到安全圈缩小到和目标圈一样大，则产生新的目标圈
20. def\_\_init\_\_(self,current\_x1,current\_y1,current\_x2,current\_y2,target\_x1, target\_y1, target\_x2, target\_y2, info): 初始信息，当前圈的坐标和目标圈的坐标
21. 流程



由于一帧游戏时间很短，所以这些操作不一定要固定先后顺序，可以看作相同优先级，于是不将流程一一画出

1. 接口

def input\_data(self,data):

self.info.operate\_queue.put(data)

def output\_data(self):

return self.info.output\_dict

input\_data函数用于输入玩家操作信息，由服务器模块调用；output\_data函数用于输出当前玩家和游戏信息，由服务器模块调用