第二届人工智能挑战赛——枪林弹雨

平台使用说明与开发注意事项

版本 v1.0rc1

平台使用说明

压缩包 `TS20Platform_Win_X86_v1.0.zip` 为运行平台

1. AI目录用于存放玩家AI代码生成的.dll文件，要求将.dll文件改名为 `AI_${team}_${number}.dll` ，其中${team}为队伍信息，同一值的会被归入同一队伍，取值范围为[0,15]，${number}为队伍中的编号，取值范围为[0,3](例: `AI_3_2.dll` 表示队伍3的2号玩家AI)。
2. `log` 目录存放平台运行时自动生成的信息，用于开发组debug工作。
3. `playback` 目录存放回放文件以及比赛结果(如果比赛完成)。
4. 以上3个文件夹内容可以随意删除。
5. 其余文件是平台运行的环境，不可随意更改。
6. 如果启动过程中出现缺少 `MSVCP140.dll` 错误，请安装提供的 `vcruntime/vc_redist.x86.exe` 。
7. 平台运行后会提示键入每次运行的AI程序数量，键入内容为数字，为保证不同AI同时运行时对资源占用的公平性，平台会限制每次同时运行的AI数小于等于本机逻辑核心数，并且将不同的AI分配到不同的逻辑核心上。键入0代表使用所有逻辑核心，键入其他大于0的数字代表使用相应数量的逻辑核心，超过实际核心数则会被设置为本机实际最大逻辑核心数。
8. 不要使用两个逻辑核心（或单核处理器）运行3个AI程序，达成此条件可能会触发“巨硬的愤怒”。
9. 场上至少有2支队伍，否则游戏会立即结束（因为吃鸡了）。

AI编写说明

压缩包 `player-project.zip` 为供选手开发使用的工程（只提供vs2017）

1. 如果出现初次编译就大量错误，可能需要修改工程中 `Windows SDK` 至现有版本，具体修改方法如下：
解决方案管理器中AI项目(右键)->属性->Windows SDK版本(修改为本机已有版本)。
2. `player.cpp` 文件中已经自带一个智障机器人AI，玩家可以在此基础上修改或重写。
3. 其他文件不可修改。
4. 工程将生成 `AI.dll` 文件，重命名移动到平台指定目录即可。
5. **务必采用Release X86模式生成AI.dll**，否则会报错。
6. 如有需要，可以通过清华云盘下载 `Visual Studio 2017` 离线安装包vs_2017。

AI工程架构

```
player-project
|----- api.h          API头文件
|----- api.cpp        API源文件
|----- base.h         DLL导出函数
|----- base.cpp       DLL导出函数
|----- comm.pb.h      通信协议(Google Protobuf)
|----- constant.h     常量表
|----- data.h         数据结构
|----- player.cpp     玩家AI（唯一可修改与提交的部分）
|----- comm           通信协议库
|----- google         通信协议头文件
```

数据结构

constant.h

```
struct XYPosition    //表示绝对位置的直角坐标
{
double x, y;
};
```

```
struct PolarPosition //表示相对位置的极坐标
{
double distance;     //距离
double angle;        //角度
};
```

data.h

```
struct Item    //物品信息
{
int item_ID;     //物品的ID
ITEM type;       //物品的种类（参见constant.h中的ITEM枚举）
PolarPosition polar_pos;     //物品的坐标（如果在地上）
int durability;              //物品的使用耐久
};
```

```
struct PoisonInfo    //电圈信息
{
XYPosition current_center;       //电圈当前中心
XYPosition next_center;          //下一电圈中心
double current_radius;           //电圈当前半径
double next_radius;              //下一电圈起始半径
int rest_frames;                 //帧数，参见下方注释
// if it's 1, rest_frames means rest frames to move to next status 为1时rest_frames代表当前缩圈完成还需帧数
// if it's 0, rest_frames means rest frames to start to move 为0时rest_frames代表距离下一次缩圈的帧数
// if it's -1, the poison ring won't move(has become a node) 为-1时代表圈已经缩为点
int move_flag;                   //标记，参看上方注释
};
```

```
struct SelfInfo      //本AI信息
{
double hp;           //当前血量
double hp_limit;     //血量上限
double move_angle;   //当前行进的绝对角度
double view_angle;   //当前视线的绝对角度
double move_speed;   //当前移动速度
VOCATION vocation;   //职业
STATUS status;       //当前状态（参看STATUS枚举）
int move_cd;         //距本次移动结束的帧数
int attack_cd;       //距可发起下一次攻击的帧数
XYPosition xy_pos;   //本AI在地图上的XY坐标
double view_width;   //当前视角
double view_distance;    //视野（与视角一起形成一个扇形为可见区域）
std::vector<Item> bag;   //本AI拥有的道具（枪、防具等等）
```

```
//注：防具自动装备
};
```

```
struct OtherInfo        //视野中其他AI的信息
{
int player_ID;      //该AI的ID
STATUS status;      //该AI的状态
double move_angle;  //该AI的行进的绝对角度
double view_angle;  //该AI的视线的绝对角度
double move_speed;  //该AI的移动速度
VOCATION vocation;  //该AI的职业
PolarPosition polar_pos;     //该AI的相对于自身的极坐标相对位置
};
```

```
struct Sound            //无线电与声音信息
{
int sender;         //无线电发送者ID或环境声音(为-1时)
int delay;          //延时，用于预估距离
SOUND type;         //声音类型（参见SOUND枚举）
int32_t parameter;  //无线电内容
};
```

```
struct PlayerInfo       //所有信息的聚合
{
int player_ID;      //本AI的ID
SelfInfo self;      //自身信息
std::vector<int> landform_IDs;  //视野中的地形的ID
std::vector<Item> items;        //视野中的物品（在地上的）
std::vector<OtherInfo> others;  //视野中的其他AI（自行辨别敌友）
std::vector<Sound> sounds;      //听到的声音与收到的无线电
PoisonInfo poison;              //电圈信息
};
```

player.cpp

```
extern XYPosition start_pos, over_pos;  //航线的起点与终点的XY坐标
extern std::vector<int> teammates;      //队友ID
extern int frame;                       //当前帧数（从0开始计数）
extern PlayerInfo info;                 //所有信息的聚合
```

常量表

- **结构体数组建议视为表格阅读**
- `AIRPLANE_SPEED`:飞机速度
- `JUMPING_SPEED`:跳伞过程中人的移动速度

```
enum STATUS //状态
{
RELAX = 0,  //无动作
ON_PLANE = 1,    //在飞机上
JUMPING = 2,     //正在跳伞
MOVING = 3,      //移动中（与move_cd相关联）
SHOOTING = 4,    //射击中（与attack_cd相关联）
```

```
PICKUP = 5,       //正在捡东西
MOVING_SHOOTING = 6,     //正在移动中射击
DEAD = 7,         //假死（可被医疗兵救活）
REAL_DEAD = 8     //真死
};
```

```
enum VOCATION
{
MEDIC = 0,        //医疗兵
SIGNALMAN = 1,    //通信兵
HACK = 2,         //黑客
SNIPER = 3,       //狙击手
VOCATION_SZ = 4,     //便于遍历（下同）
};
```

- PS:先略去能顾名思义的内容
- `vocation_property`:存放职业常量的结构体
- `VOCATION_DATA`:职业信息表格
- `ITEM_TYPE`:道具的类型
- `ITEM_MODE`:道具的使用模式（值为 `ITEM_MODE_SZ` 代表无模式）
- `ITEM`:道具种类枚举
- `item_property`:存放道具常量的结构体
- `ITEM_DATA`:道具信息表格
- `SOUND`:声音/无线电种类枚举
- `sound_property`:存放声音/无线电常量的结构体
- `SOUND_DATA`:声音/无线电信息表格
- `BLOCK_SHAPE`:地形中的物体（树、房子等）的形状
- `BLOCK_TYPE`:地形中的物体枚举
- `block`:物体信息的结构体（其中的x0,y0等参数与物体的形状有关，参见代码注释）
- `AREA`:地形块枚举
- `AREA_DATA`:每一个地形块的组成方式（每个地形块100×100范围上的物体，位置均为相对位置）
- `MAP_SZ`:数组 `MAP` 的大小
- `MAP`:描述地图的构成（由10×10个地形块构成，排序为先x轴从小到大，再y轴从小到大，整个地图为*1000×1000）
- `circle_property`:存放电圈信息的结构体
- `CIRCLE_SZ`:数组 `CIRCLE_DATA` 大小
- `CIRCLE_DATA`:电圈数据表格

API

- 所有的API均在命名空间ts20下，以避免与标准库冲突

```
//跳伞，只能在第0帧时使用，未调用的AI将被平台自动关闭
//参数：选择的职业与跳伞地点的XY坐标
void parachute(VOCATION role, XYPosition landing_points);
```

```
//射击与使用道具
//参数：使用的道具/枪的枚举，相对角度，特殊参数（某些特殊物品需要）
void shoot(ITEM item_type, double shoot_angle, int parameter =-1);
```

```
//移动
//参数：前进方向与视角的相对角度（相对于当前视角）
```

```
void move(double move_angle, double view_angle);
```

```
//拾取
//参数：地上的物品ID，需要在一定范围内才能成功（distance<1?）
void pickup(int target_ID, int parameter = -1);
```

```
//发送无线电
//参数：接收者ID，信息（只能使用低29位）
void radio(int target_ID, int msg);
```

```
//更新信息（player.cpp中的info与frame）
//阻塞式，如果没有新的信息到来就一直等待
//play_game函数中至少出现一次，否则无法收到最新数据
void update_info();
```

```
//更新信息，同上
//非阻塞式，如果没有新信息就直接返回false
//否则解析新信息并返回true
bool try_update_info();
```

```
//将地形ID转换为具体物体的结构体
//返回的block中的结构中的位置信息是绝对位置
block get_landform(int landform_ID);
```

名词解释

- 帧数:每个AI每次执行一定时间，称为1帧，相当于整个连续时间的离散化表示
- 绝对位置:在整个地图上的XY坐标
- 相对位置:极坐标，以当前人的视野为极坐标角度0值处，逆时针方向增大，角度取值范围在0~360

平台简介

- 平台采用伪实时设计，在逻辑返回的同一时刻信息的基础上，所有AI并行运行（并行数受使用的逻辑核心数限制），多个批次之间串行运行，保证所有的AI运行相近的时间，如果在时间限制内函数没有返回，AI将被挂起，等待下一次执行。每一帧后，平台会收集AI调用API的数据反馈给逻辑更新场景信息，并反馈给平台，当AI调用update系列函数后更新信息。

附

1. 有任何问题请联系开发组解决

最后更新于2019年3月28日