2019年智能体大赛SDK说明文档

阅读提示：本文档仅供已经了解游戏规则，并希望编写AI的选手参考。如果您只是希望了解本游戏，请关闭本文档，出门左转，以免对本游戏丧失兴趣。

前言

欢迎您参加 2019 年智能体大赛。在阅读本文档之前，请先阅读游戏规则说明文档及逻辑参考文档。

如果您阅读到此处，想必您已经熟悉逻辑运作的模式，并且做好编写 AI 并参赛的准备。为了您的方便，我们向您提供这份指导教程，使您能了解逻辑与 AI 的交互方式，以及 SDK 的使用方法。

如您认为本文档中有明显的错误或哪里语焉不详，欢迎您在智能体大赛 QQ 群中提出，或者联系384520721@qq.com。我们将尽快给出回复。

文件结构说明

playerAI.h & playerAI.cpp

仅包含函数

```cpp
void playerAI();
```

选手需要完成在player.cpp中完成该函数，之后说明如何完成。

const.h

储存所有可能会用到的常数。可以使用

```cpp
using namespace CONST;
```

或者

```cpp
CONST::变量名
```

进行调用。

geometry.h & geometry.cpp

内含基本几何结构和可能会用到的计算几何库。函数列表及参数说明可直接查看头文件。

logic.h & logic.cpp

重要的数据类。选手需要关注Human类、Fireball类、Meteor类、Crystal类、Map类以及Logic类。前五类都是基本数据类型，之后会具体说明，同时可参见logic.h的注释。Logic类包含了当前地图全部的信息并提供操作接口。在每帧开始时(即playerAI函数被调用)，Logic会将所有数据更新至最新。决策结束后(即playerAI函数结束返回)，Logic会将选手在playerAI中进行的操作记录并发送。Logic使用单例模式，确保整个程序只有一个Logic类并可以通过静态函数获取单例指针。使用时请使用

```cpp
Logic::Instance()
```

获取Logic单例指针，以获取本帧所有数据并调用操作接口进行决策。

注意，所有数据均为public数据，请谨慎修改。如有修改，下一次决策时将会被最新的数据覆盖。

main.cpp

与服务器进行通信，选手可以忽略这个文件。

重要数据类型

我们通过定义若干类型储存数据。以下介绍这些数据类型。介绍不会过于详细，了解更多请参见源代码。

所有数据类型均定义在Logic.h。

Human

人物类，包含编号、位置、生命值、剩下的陨石数量、陨石术剩余冷却时间、剩余闪现数量、闪现剩余冷却时间、开火剩余冷却时间、死亡剩余时间、无敌时间。

Fireball

火球类，包含位置、朝向、来自哪个人。

Meteor

陨石类，包含位置、剩余存在时间、来自哪个人

Crystal

水晶类，包含位置、归属(指被扛起的人物编号)、所属势力。

Map

地图类，包含宽、高、势力个数、每个实例控制的人数、每个人的出生地、每个势力的水晶初始位置、每个势力的水晶搬运目标位置、每个加分道具的位置、地图的像素信息、游戏总时间。

Logic

总数据类，包含所有需要的数据和会用到的接口。

数据包括，本帧的帧数、自己势力的编号、地图、所有人物、所有火球、所有陨石、所有水晶、加分道具是否存在。

接口包括

```
cpp
void move(int num, Point p);//指定你控制的第num个人移动到p位置
void shoot(int num, Point p);//指定你控制的第num个人向p位置发射火球
void meteor(int num, Point p);//指定你控制的第num个人向p位置释放陨石术
void flash(int num);//指定你控制的第num个人本次移动改为闪现
void unmove(int num);//取消你控制的第num个人的移动指令
void unshoot(int num);//取消你控制的第num个人的射击指令
void unmeteor(int num);//取消你控制的第num个人的发射陨石指令
void unflash(int num);//取消你控制的第num个人的闪现指令
```

注意，这里你控制的第num个人实际上是humans中的humans[j*n+num]，其中j指你的势力标号，n指总势力个数。

新增debug接口如下

```
cpp
void debug(string msg);//设置debug信息，会覆盖本帧之前设置的信息
void debugAppend(string amsg);//追加debug信息，不会覆盖之前的信息
```

每帧将重置debug信息，每帧的debug信息不能超过1024字节，如果需要加长信息，请修改main.cpp(但不建议过长，否则可能导致通信中断)。

注意:请不要在程序中进行任何的stdio，否则将导致通信中断。

SDK运行逻辑

本节介绍与选手有关的SDK运行逻辑，详见源代码。

第一次调用playerAI函数一定是第1帧，此时Logic实例中已经获取好第一帧的所有信息。选手通过完成playerAI函数，利用Logic的所有信息以及其他常数和计算几何函数，或者自己定义的其他数据类型与函数(允许选手自行添加其他文件，但选手可能需要更改makefile文件)，进行决策，通过调用上述介绍的Logic内的8个接口进行操作。

playerAI结束后，Logic中的ope变量已经获取好本帧选手的所有操作，之后会通过main.cpp中的相关函数发送给服务器，然后等待下一帧的到来。下一帧到来之后将数据更新到Logic实例中，并调用playerAI函数，重复循环。

允许选手使用多帧进行决策，并保证下次决策开始前获取的是最新信息。例如，第i帧开始时选手开始决策(playerAI函数开始调用)，在第i+k帧开始之后、i+k+1帧开始之前决策结束(playerAI函数结束返回)，那么选手的操作将在i+k+1帧产生作用，i+k+1帧开始时继续调用playerAI函数，此时数据为i+k+1帧的数据。选手决策期间(i+1帧到i+k帧)选手所有人物处于静止状态。