

开发指南

运行游戏的方法

首先运行服务端，接着运行两个客户端（bot也算是一个客户端）。对于服务端和客户端的功能描述，请参考下文“Client-Server运行流程”

```
# launch server
cd bin
./server # if run into permission denied problem, run `chmod +x server`
first

# launch bot
cd bin
./silly-bot # if run into permission denied problem, run `chmod +x server`
first

# launch client, take python client as example
cd client/python
python main.py
```

注意，上述命令是运行三个可执行文件的命令，由于shell执行程序的时候会被当前程序占用（除非你用 & 放在后台运行），因此三个程序并不能在同一个终端中运行。正确的做法是开三个终端分别运行。如果你神通广大，你也可以使用tmux等分屏在同一个终端中分出三个部分分别运行。

Client开发指南

Client-Server运行流程

这部分主要是给没学过计算机网络的同学看的

名词解释

- client:客户端，在本例中是一个程序，能够将用户的信息（一些字节流）发送给server（服务端）。
- server:服务端，在本例中也是一个程序，能够将游戏的信息（当前地图或人物的状态等）发送给client。

本游戏是一个多人联机游戏，每一个玩家运行一个client并与server建立通信连接，玩家通过client将想要执行的动作（例如向上走一步或放置炸弹）发送给server，server处理玩家的动作并更新游戏的状态（人物的位置等），并将新的游戏状态回传给client,玩家可以在根据接收到的游戏状态信息执行下一步的指令。

选手需要做些什么

在本次比赛中，选手需要编写代码，根据client接收到的游戏状态信息进行决策，并将要让角色做出的动作（例如向上走一步或放置炸弹）回传给server。

注意，选手并不需要编写server端，因为游戏的机制是固定的。选手只需要修改client端，接收游戏的状态信息并让做出相应的动作即可。

客户端代码

本次组委会提供C++、Python、Go三种语言的sdk方便开发。本仓库仅包含Python语言编写的客户端代码，如果需要使用C++或者Go语言进行开发，请参考下面的链接。

C++客户端代码：<https://gitee.com/chenxuan520/seedcup-cppsdk>

Go客户端代码：<https://github.com/chenxuan520/seedcup-gosdk>

传输协议

客户端和服务端通信需要遵循一定的协议，为了便于选手debug，采用json序列化及反序列化。在python客户端中已经实现了通信的协议，理论上选手可以直接按照客户端提供的接口即可。

总的协议体如下：`type`字段为1表示InitReq，`type`字段为2表示ActionReq，`type`字段为3表示ActionResp，`type`字段为4表示GameOverResp。`data`字段则包含具体的信息。

```
{  "type": 1,  "data": {...}}
```

Request协议

InitReq

- InitReq的packet样例

```
{  "type": 1,  "data": {"player_name": "seed soldier"}}
```

- InitReq由client向server发送，请求server添加一名玩家

ActionReq

- Action请求告知服务端客户端要进行的具体行动。

```
{  "playerID": 1,  "actionType": 2 }
```

- actionType

```
enum ActionType {  
    SILENT = 0, // 静止不动  
    MOVE_LEFT = 1,  
    MOVE_RIGHT = 2,  
    MOVE_UP = 3,  
    MOVE_DOWN = 4,  
    PLACED = 5, // 放置炸弹或者道具 相当于空格  
};
```

- ActionReq的packet样例

```
{"type": 2, "data": [{"playerID": 0, "actionType": 0}, {"playerID": 0, "actionType": 1}]}
```

这里的data字段是一个列表，每一个元素代表一个动作，一个回合内可以在行动速度的限制下执行多个动作

Response协议

ActionResp

ActionResp会返回整个地图信息，选手可以利用这些信息训练模型或进行决策。

```
{
  "player_id": 0,    // (自己) 玩家的id
  "round": 20,       // 当前的回合数
  "map": []          // 地图信息
}
```

- map是一个列表,内含多个小方格,每个小方格字段样例如下

```
{
  "x": 0,            // 小方格的x坐标
  "y": 0,            // 小方格的y坐标
  "last_bomb_round": 2 // 上一次被炸的回合
  "objs": []         // 小方格上的物体
}
```

- 其中objs是一个列表,代表当前小方格上的物体,一个小方格可能含有0个或多个物体。objs中每个元素示例如下

```
{"type": 1, "property": {}}
```

- 其中type字段表明物体的类型，其对应关系为

type	物体类型	备注
1	Player	玩家
2	Bomb	炸弹
3	Block	障碍
4	Item	道具

- player的property字段如下

```
{
  "player_id": 0,      // 玩家id
  "alive": true,      // 是否存活
  "hp": 100,          // 血量
  "shield_time": 2,    // 护盾剩余回合数
  "invincible_time": 2, // 无敌回合数
  "score": 10,         // 当前总分数
  "bomb_range": 5,     // 炸弹爆炸范围
  "bomb_max_num": 2,   // 炸弹数量上限
  "bomb_now_num": 1    // 当前剩余炸弹
  "speed":2            // 玩家的移动速度
}
```

- Bomb的property字段如下

```
{
  "bomb_id": 0,      // 炸弹id
  "bomb_range": 5,   // 炸弹范围
  "player_id": 2     // 炸弹放置人
}
```

- Block的property字段如下

```
{
  "block_id": 0,      // 障碍物的id
  "removable": true   // 该障碍是否可被清除
}
```

- Item的property字段如下

```
{
  "item_type":1
}
```

- 其中item_type与道具对应的关系如下

item_type	类型	备注
1	BOMB_RANGE	增加炸弹范围
2	BOMB_NUM	增加炸弹数量上限
3	HP	增加生命

item_type	类型	备注
4	INVINCIBLE	获得无敌效果
5	SHIELD	获得护盾

- ActionResp的packet样例

```
{
  "data": {
    "map": [
      // (map列表中的每个元素由多个{}组成，每一个{}代表地图上的一个小方块)
      {
        "last_bomb_round": -1,
        // (objs列表中的每个元素由0个或多个obj部分组成)
        "objs": [
          {
            "property": {
              "alive": true,
              "bomb_max_num": 2,
              "bomb_now_num": 0,
              "bomb_range": 1,
              "hp": 1,
              "invincible_time":
0,
              "player_id": 3,
              "score": 0,
              "shield_time": 0,
              "speed":2
            },
            "type": 1
          },
          {
            "last_bomb_round": 36,
            "objs": [
              {
                "property": {
                  "item_type": 2
                },
                "type": 4
              }
            ],
            "x": 11,
            "y": 14
          },
          {
            "last_bomb_round": -1,
            "objs": [
              {
```

```

        "property": {
            "bomb_id": 8,
            "bomb_range": 1,
            "player_id": 2
        },
        "type": 2
    },
    {
        "x": 14,
        "y": 13
    },
    {
        "last_bomb_round": -1,
        "objs": [
            {
                "property": {
                    "block_id": 320,
                    "removable": true
                },
                "type": 3
            }
        ],
        "x": 14,
        "y": 11
    },
    {
        "last_bomb_round": -1,
        "objs": [],
        "x": 14,
        "y": 2
    }
],
"player_id": 3,
"round": 48
},
"type": 3
}

```

GameOverResp

当游戏结束的时候服务端会返回GameOverResp，包括每一个玩家的id及其对应的得分，样例如下

```

{
    "type": 4,
    "data": {
        "scores": [
            {"player_id": 2,
            "score": -11630},
            {"player_id": 3,
            "score": 10830}],
        "winner_ids": [3]
    }
}

```

```
}  
}
```