1.文件结构：

Hex/

├── Hex/

│ ├── \_\_init\_\_.py

│ ├── settings.py

│ ├── urls.py # 主路由配置

│ └── wsgi.py

│ └── asgi.py # 新增ASGI配置

└── Hexgame/

├── migrations/ # 数据库迁移文件

├── static/

├── css/

├── image/

├── js/

├── templates/

├── html/

├── utils/

├── utils.py # 工具函数

├── Algorithm.py # 算法函数

├── Hexmodel.py # 模型函数

├── Selfplay.py

├── models.py # 数据模型

├── serializers.py # 序列化器

├── routing.py # 新增WebSocket路由

├── consumer.py # 新增WebSocket消费者

├── views.py # API视图

└── urls.py # 子路由配置

2.前后端交互：

基础URL：ws://localhost:8000/

| 接口类型 | 连接地址 | 动作（action） | 请求数据 | 响应类型 | 响应数据结构 |
| --- | --- | --- | --- | --- | --- |
| 创建游戏 | ws/game/new/ | create\_game | {"action": "create\_game",  "mode": "HUMAN\_AI" } | game\_created | {"type": "game\_created", "data": { "game\_id": "123", "redirect": "/ws/game/123/" } } |
| 加入游戏房间 | ws/game/{id}/ | 无（自动连接） | 无 | game\_state | { "type": "game\_state", "data": { "board": [...], "player\_turn": "human", "mode": "HUMAN\_AI", "win\_probability": { ... }, "winner": null } } |
| 玩家移动 | ws/game/{id}/ | move | { "action": "move",  "x": 0, "y": 0 } | game\_update | { "type": "game\_update", "data": { "board": [...], "player\_turn": "AI", "win\_probability": { ... }, "winner": "human" } } |
| AI移动 | ws/game/{id}/ | ai\_move | { "action": "ai\_move" } | game\_update | 同上 |
| 悔棋 | ws/game/{id}/ | undo | { "action": "undo" } | game\_update | 同上 |
| 重置游戏 | ws/game/{id}/ | restart | { "action": "restart" } | game\_update | 同上 |
| 错误处理 | 所有接口 | 无效动作/参数 | 任意无效请求 | error | { "type": "error", "data": { "message": "错误描述" } } |

//所有数据结构：   
{

"id": 123, // 游戏ID

"board": [ // 11x11棋盘状态

[0, 1, -1, 0, ...], // 0=空, 1=红方, -1=蓝方

[0, 0, 0, ...],

...

],

"player\_turn": "human", // 当前回合玩家 ["human", "AI", "AI\_1", "AI\_2"]

"human\_color": "red", // 玩家颜色 ["red", "blue"]

"last\_moves": [ // 最后落子位置标记（数字表示步数）

[0, 1, 2, 0, ...],

[0, 0, 0, ...],

...

],

"moves\_history": [ // 历史动作记录

{

"player": "human",

"x": 1,

"y": 2,

"step": 1,

"color": "red"

},

...

],

"winner": null, // 获胜者 ["human", "AI", "AI\_1", "AI\_2", "draw"]

"win\_probability": { // 实时胜率预测

"human": 0.6,

"AI": 0.4

},

"mode": "HUMAN\_AI" // 游戏模式 ["HUMAN\_AI", "AI\_AI"]

}

1. 关键说明

连接流程：

通过 /ws/game/new/ 创建游戏 → 获取 game\_id → 重定向到 /ws/game/{id}/。

模式类型：

HUMAN\_AI：玩家与AI对战。

AI\_AI：AI自动对战。

棋盘坐标：

x 和 y 范围是 0-10（对应11x11棋盘）。

胜率计算：

在 HUMAN\_AI 模式下，win\_probability 包含人类和AI的胜率。

在 AI\_AI 模式下，包含两个AI的胜率。

4.需要安装的库

pip install websockets requests

pip install channels channels-redis

pip install daphne

Pip install Django

5.Django框架运行指令：  
本地测试：  
在terminal终端输入 daphne Hex.asgi:application 运行服务器  
在另一个终端输入 python hex\_test.py可以启动测试脚本

其他同学通过 WebSocket 访问你的本地服务器步骤：  
1)在项目的终端输入：daphne -b 0.0.0.0 -p 8000 Hex.asgi:application

-b 0.0.0.0：允许外部设备访问。

-p 8000：指定端口（默认为 8000）  
  
2)获取IP地址

Windows：在命令提示符输入 ipconfig，查找 IPv4 地址（如 192.168.1.100）。

Mac/Linux：在终端输入 ifconfig，查找 inet 地址（通常以 192.168.x.x 或 10.x.x.x 开头）。

3)配置防火墙放行端口

Windows：

打开 控制面板 → 系统和安全 → Windows Defender 防火墙 → 高级设置。

新建入站规则，允许 TCP 端口 8000。  
Mac：  
sudo pfctl -f /etc/pf.conf # 编辑后重新加载规则

4)确保同学在同一局域网

其他同学必须和你的电脑连接到 同一个局域网（如同一 Wi-Fi 或路由器下的网络）

5)提供访问地址

其他同学需使用以下格式的 WebSocket 地址：  
# 创建游戏

ws://[你的本地IP]:8000/ws/game/new/

# 加入游戏房间

ws://[你的本地IP]:8000/ws/game/[game\_id]/