**实验5 飞行棋游戏：以太坊多方参与合约**

**【实验介绍】**

飞行棋是一种深受人们喜爱的棋类游戏，传统的飞行棋支持2~4人同时进行游戏，玩家通过掷骰子数决定自己的棋子在棋盘上行进的步数，以全部棋子最先到达终点的游戏者为最终的胜利者。在区块链环境中，我们可以通过设计相应的智能合约实现一个有趣的支持多方参与的去中心化的飞行棋游戏。

通过上一实验，我们了解了以太坊依赖环境及相关库，学习了简单合约编写、编译和部署方法。在本实验中，我们将使用Solidity语言来实现具有应用价值的智能合约，并将其署到Ganache中建立的个人区块链系统上，再通过Python远程rpc来调用这个智能合约。本实验的设计目的是帮助读者掌握智能合约的编写方法，同时熟悉通过Python调用以太坊rpc接口的过程。

（1）Ganache

在本实验中，我们使用Ganache建立一个个人以太坊区块链作为本实验的区块链环境。Ganache是一个以太坊可视化工具，我们可以通过Ganache快速查看用户的状态，包括其地址、私钥、交易以及账户中的以太币余额，还可以查看其内部区块链的日志输出。可以使用Ganache运行测试、执行命令、检查状态，同时控制区块链的运行方式。

在Python中与以太坊建立连接需要导入web3包，主要用到的函数为Web3(Web3.HTTPProvider(‘http://ip:port’)、w3.eth.contract()。

（2）Solidity

Solidity语言是一种面向合约的高级编程语言，其风格与JavaScript很像，但实际上有很多不同的地方。Solidity语言是针对以太坊虚拟机设计的，用于在以太坊区块链网络上实现智能合约，它能够在以太坊的去中心化环境中执行。Solidity语言是静态类型语言，支持继承、库和复杂的用户定义类型， Solidity语言中合约（contract）类似于其他面向对象语言中的“类”的定义，其中包含了数据以及对这些数据进行操作的方法，每个合约中可以包含状态变量、函数、函数修饰器、事件、结构类型和枚举类型的声明，合约可以从其他合约继承而来。

Solidity 中的映射类型将本实验中重点使用，Solidity 中映射类型声明的语法与 C/C++和Java中的映射不同，Solidity中的映射类型在声明时的形式为 ：

mapping(\_KeyType => \_ValueType)

其中 \_KeyType 可以是除了映射、变长数组、合约、枚举以及结构体以外的几乎所有类型。 \_ValueType 可以是包括映射类型在内的任何类型。通常情况下，映射可以视作哈希表，映射与哈希表的不同点在于，在映射中，实际上并不存储 key，而是存储它的 keccak256 哈希值，从而便于查询实际的值。如果将映射声明为 public，合约外部的函数将可以调用Solidity 创建的getter函数来访问该映射。 getter的参数为\_KeyType，getter 的返回值为\_ValueType。\_ValueType 本身也可以是一个映射，这时在使用 getter 时将需要递归地传入每个 \_KeyType 参数。

Solidity项目主要靠Solidity编译器产出，Solidity编译器将代码编译成字节码以及该项目的二进制接口标准。应用程序二进制接口（Application Binary Interface，ABI），是两个程序模块之间的接口，定义了机器指令中数据结构和函数是如何被访问的。但是，ABI与API不同，API是高级语言编写的可供程序员阅读和使用的源代码接口，而ABI是在机器指令层面上的编码、解码以及传送数据的方式。这里的字节码和程序二进制接口在上一实验中已经提到，在本实验中体现为编译合约所生成的.abi文件和.bin文件。

本次实验的Python代码已经提供，Solidity代码中的play函数需要读者实现，本实验的代码框架如图5-1所示。其中，.abi文件是合约编译生成的应用程序二进制接口；.bin文件为合约编译生成的字节码，即运行在以太坊虚拟机上的代码；.py文件的功能是连接到区块链与区块链进行交互；.sol文件是需要编写的合约文件。

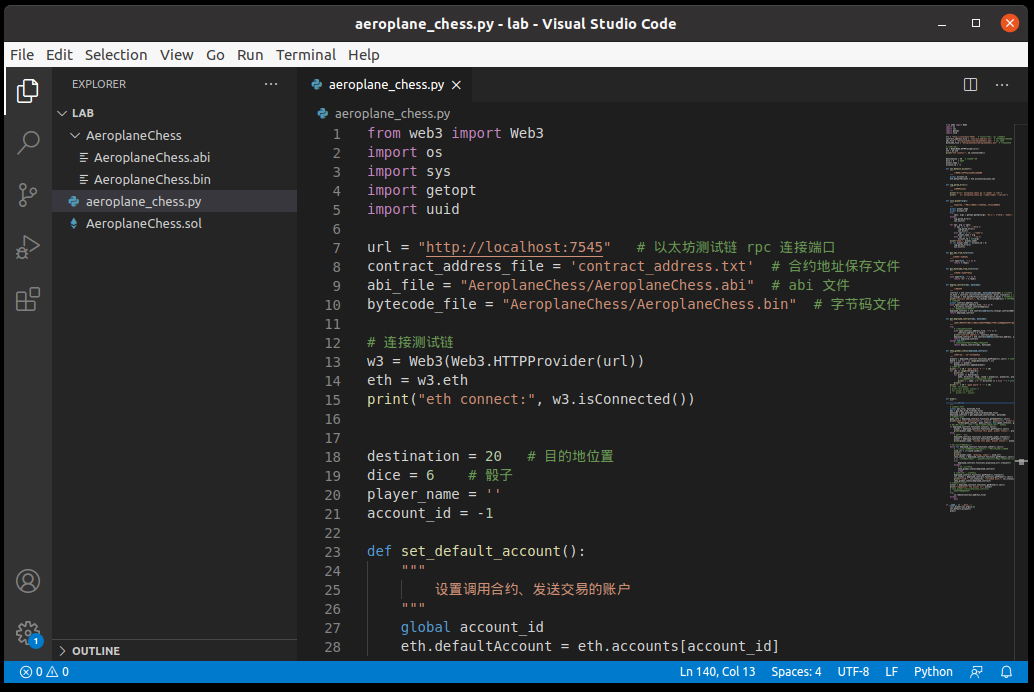


图 5-1 代码结构

**【实验要求】**

(1) 熟悉Python通过以太坊rpc接口部署并调用合约的过程

(2) 掌握solidity的结构体、数组、mapping等数据结构

(3) 掌握require函数及基本代码的编写方法

**【实验准备】**

1. **安装相关依赖包**
2. 安装pip3

|  |
| --- |
| **sudo** apt install python3-pip **-**y |

1. 使用pip3安装web3 python包

|  |
| --- |
| pip3 install web3 |

1. 安装ipython3

|  |
| --- |
| **sudo** apt install ipython3 -y |

1. **使用ganache开启以太坊测试链**

ganache是一个以太坊可视化工具，可以快速启动以太坊测试链，工具包已下载在dase目录中，在终端输入以下命令启动gannache：

|  |
| --- |
| **cd** dase  **./**ganache-2.5.4-linux-x86\_64.AppImage **&** |

点击QUICKSTART快速启动一个以太坊测试链。

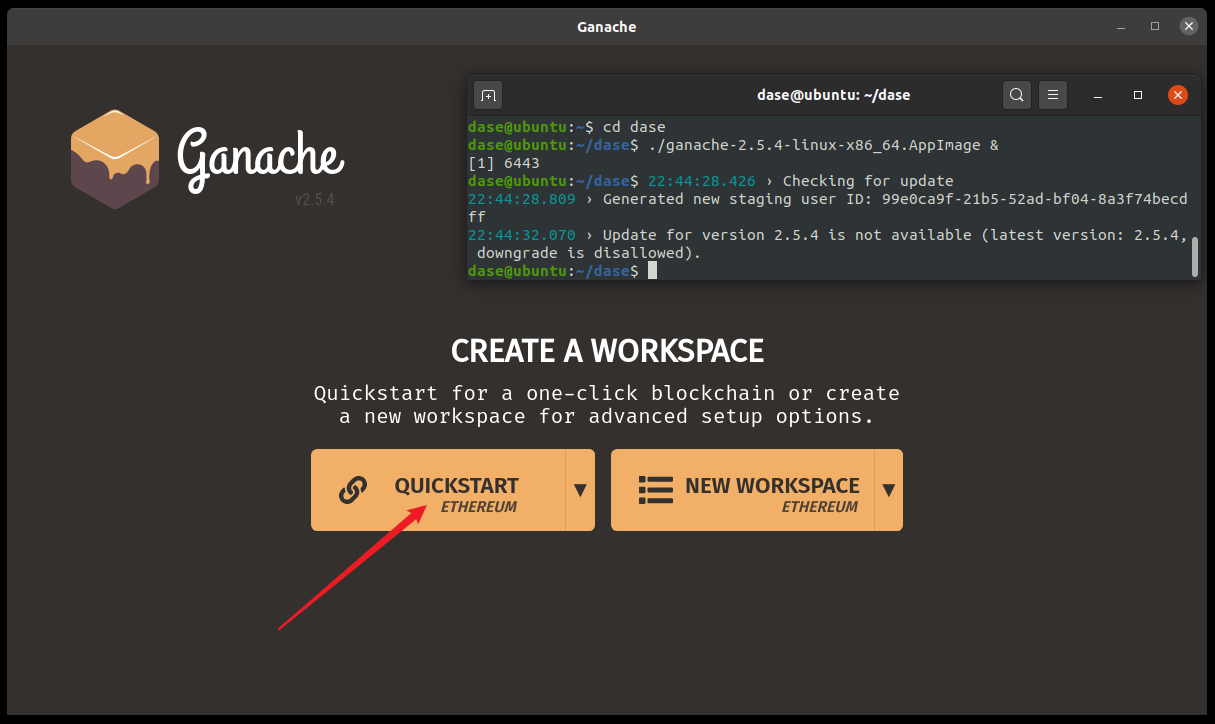


图 5-2 QUICKSTART启动以太坊测试链

如下图所示，创建了一个以太坊测试网络，默认创建十个拥有100以太币的账户，远程rpc调用端口默认为7545，ganache工具还可以查看区块和交易信息。

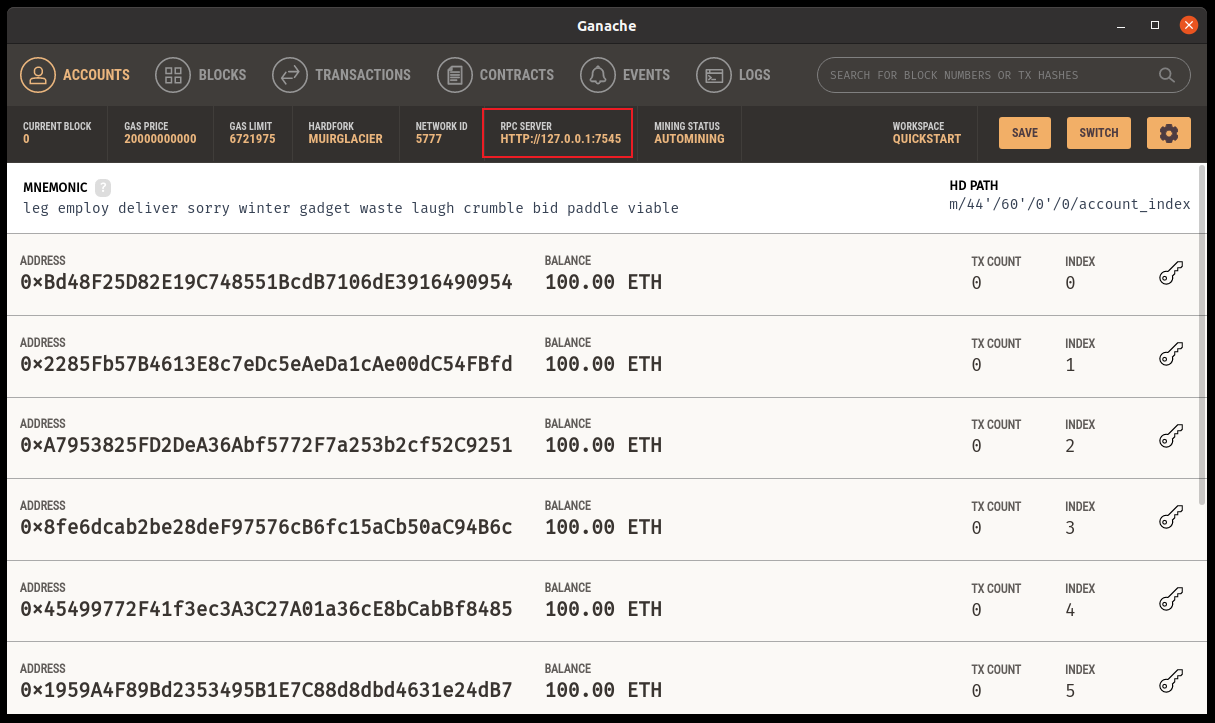


图 5-3 以太坊交易信息

1. **Python连接以太坊测试链**

打开终端，输入ipython3进入Python交互环境，输入以下代码测试连接情况：

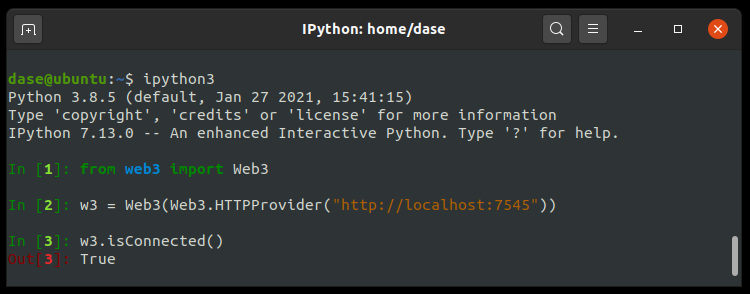


图 5-4 Python测试连接情况

1. **飞行棋游戏Python代码设计介绍**

考虑到本实验的重点在于智能合约的设计，我们实现了相应的Python代码，实验会提供相应的aeroplane\_chess.py，以太坊的多方账户参与这个简化版飞行棋游戏，最终会实现如下图的功能：

首先，我们查看以下测试链的信息，我们将使用账户0、1、2作为参与者，现在他们的以太币都是100eth：

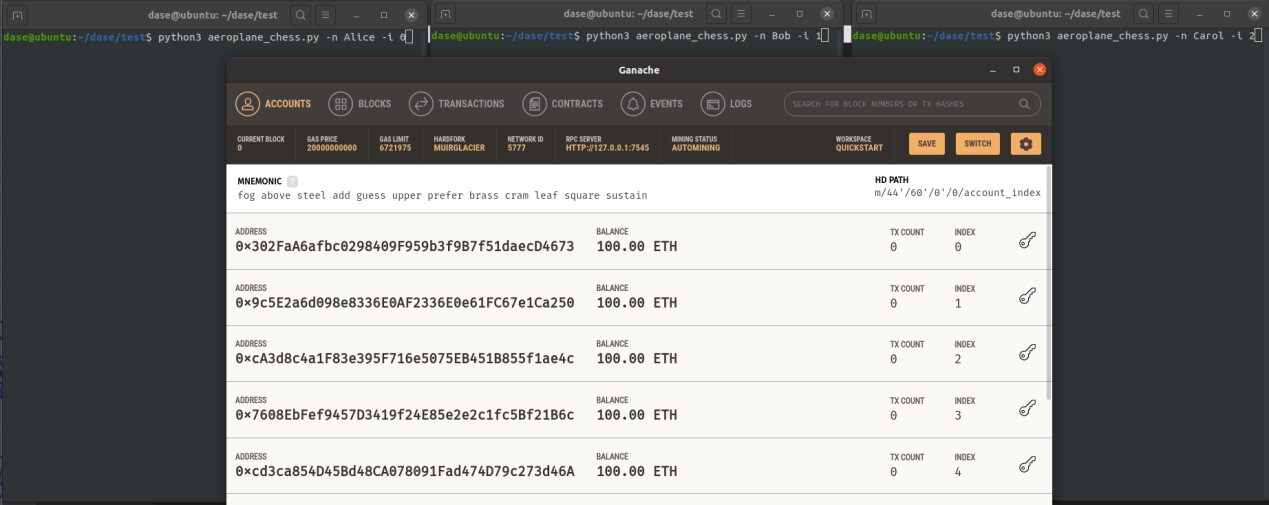


图 5-5 游戏开始前玩家以太币数量

开始游戏如下图所示，Alice、Bob、Carol三位玩家参与游戏，他们的飞机开始处于0位置，然后他们输入不同且随机的字符串，在合约中会被处理为随机骰子的值，开始向终点进发，下图中名称右方的箭头为其方向（v或^）,数字为当前骰子的步数（已走完），r+数字为当前用户游玩的轮次（因为参与时间一致轮次将会相同）。

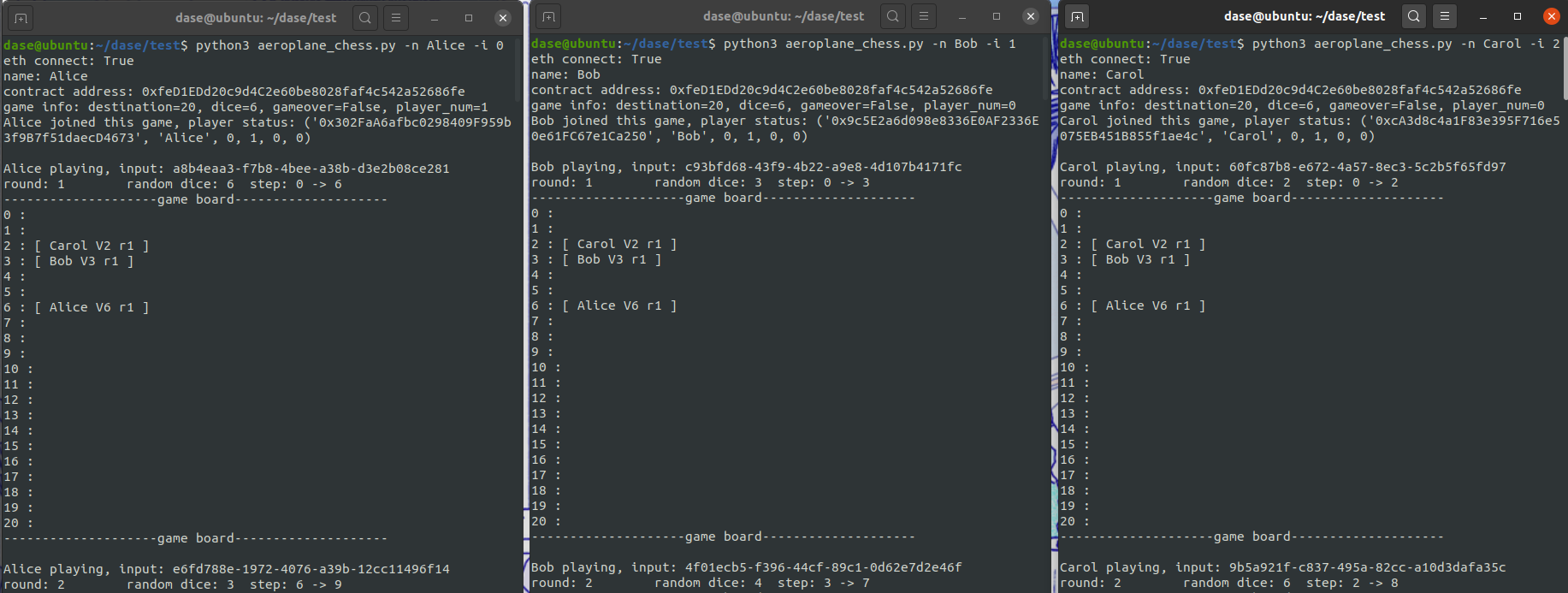


图 5-6 游戏截图1

下图右方玩家Carol回到0是因为被Alice的飞机撞回到起点，因为编码原因暂时无法提供更细节的反馈。

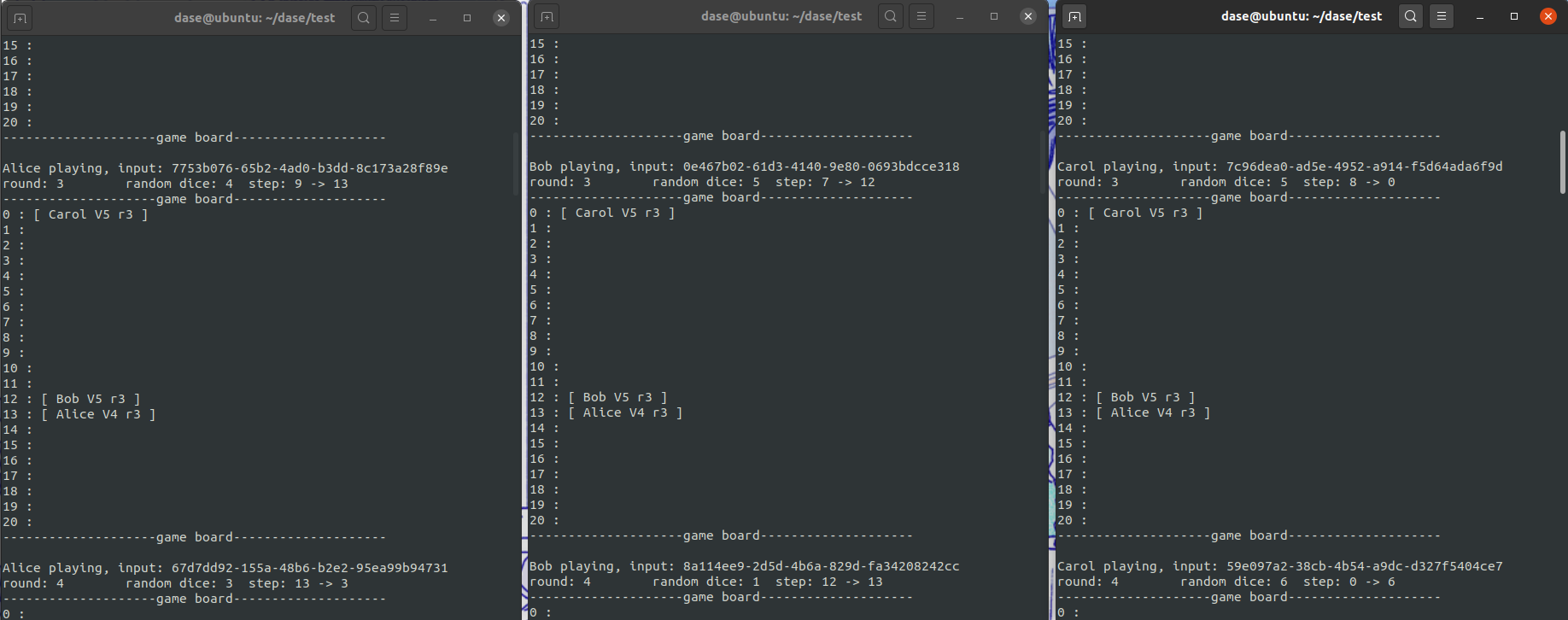


图 5-7 游戏截图2

最终，Bob的飞机飞到了终点，赢得了本次游戏。

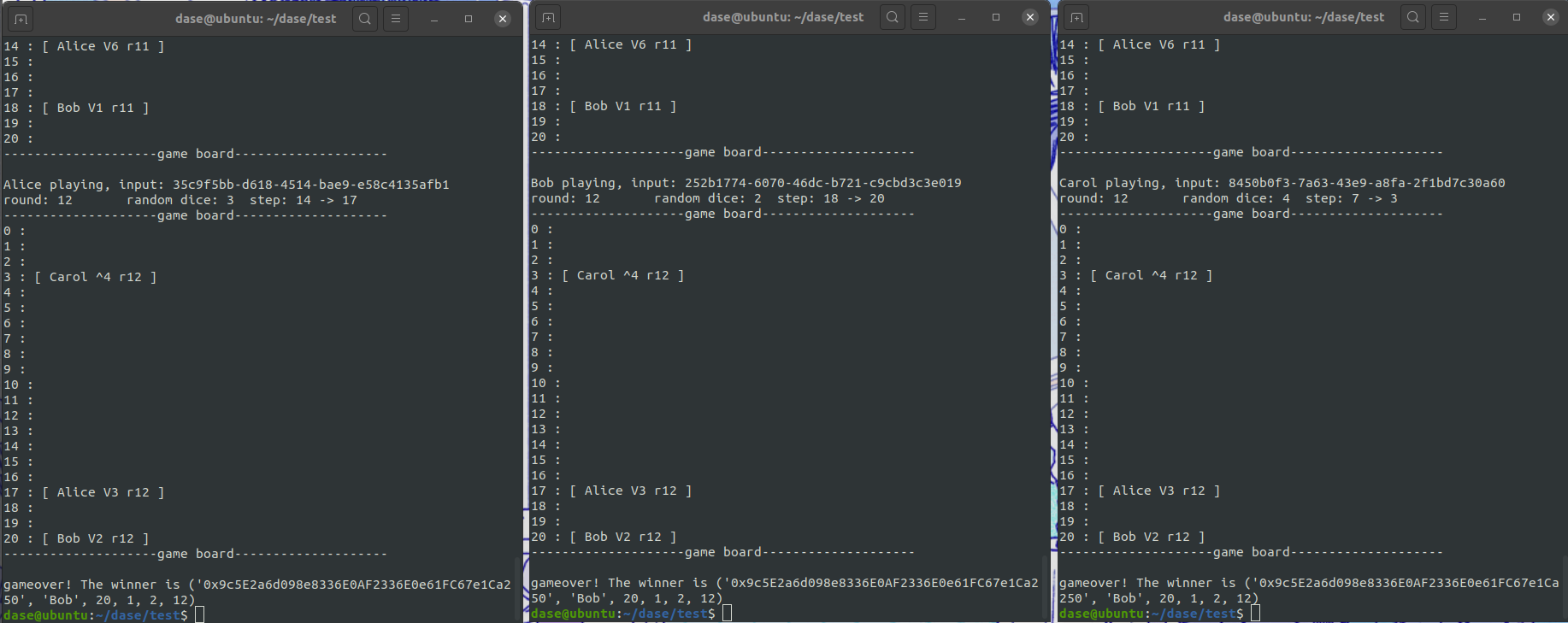


图 5-8 游戏截图3

此时我们回看ganache可视化工具，发现三个账户都消耗了一定量的以太币发送交易执行合约代码。

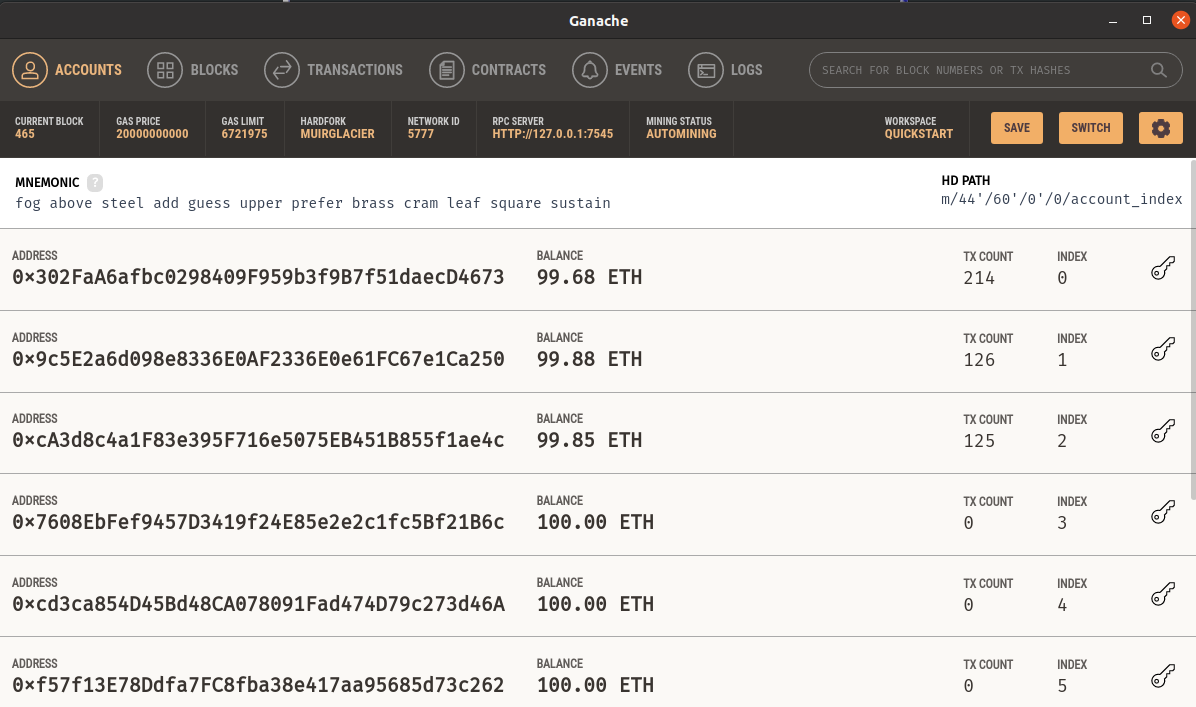


图 5-9 游戏结束后玩家以太币数量

以太坊也产生了一定量的区块去打包这些交易：

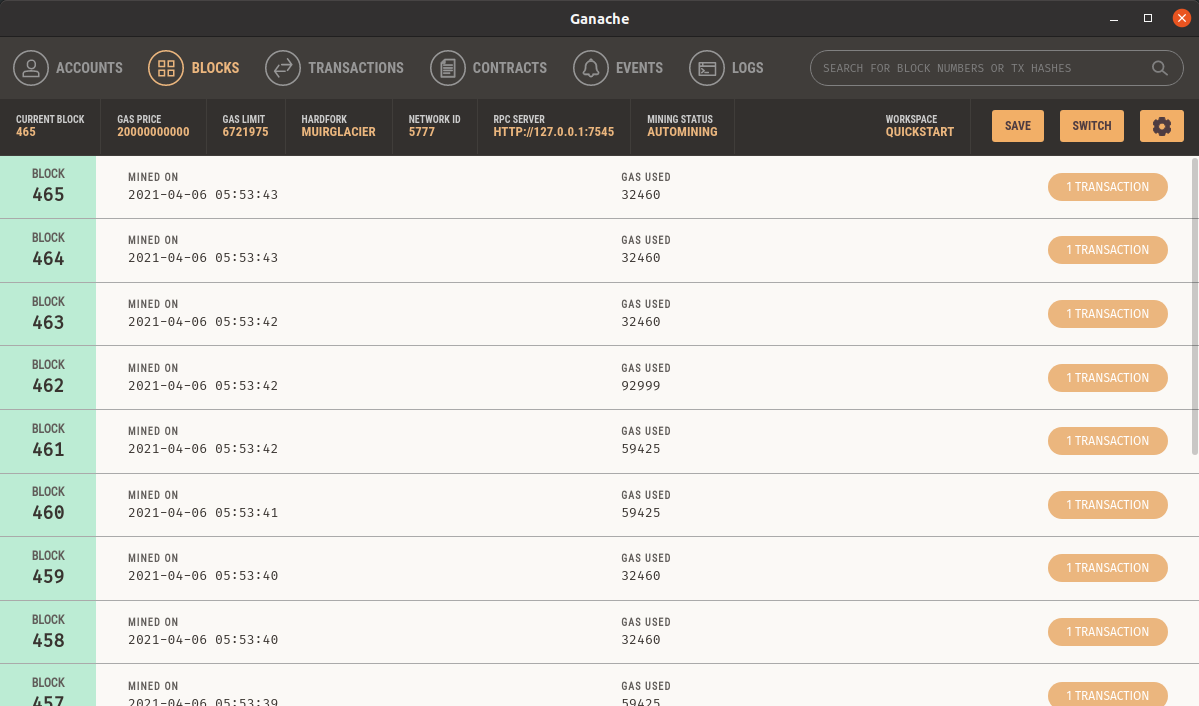


图 5-10 游戏产生的区块

1. **功能函数介绍**
2. 导入包及相应配置

目的地、骰子和玩家人数可以自行设定，但必须都是正数。注意abi文件和bytecode文件的位置。



图 5-11 python中定义路径和参数

1. 通过命令行获取游戏玩家及相应id

id对应于区块链账户数组的索引，即eth.accounts[id]。



图 5-12 python中init\_player函数

1. 从文件中读取abi和bytecode并部署合约

本次实验编写完合约，使用solc编译后，配置相关文件路径，即可通过Python自动读取文件内容并部署，减少复制粘贴的步骤。



图 5-13 deploy\_contract函数

1. 保存合约地址，如果文件及地址有效，就读取旧有合约，继续仍在进行的游戏

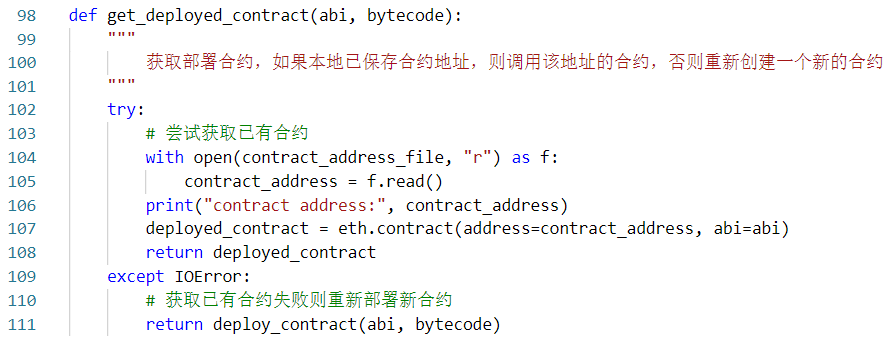


图 5-14 get\_deployed\_contract函数

1. play进行游戏函数，该游戏可中断后继续进行，详细信息请看注释：

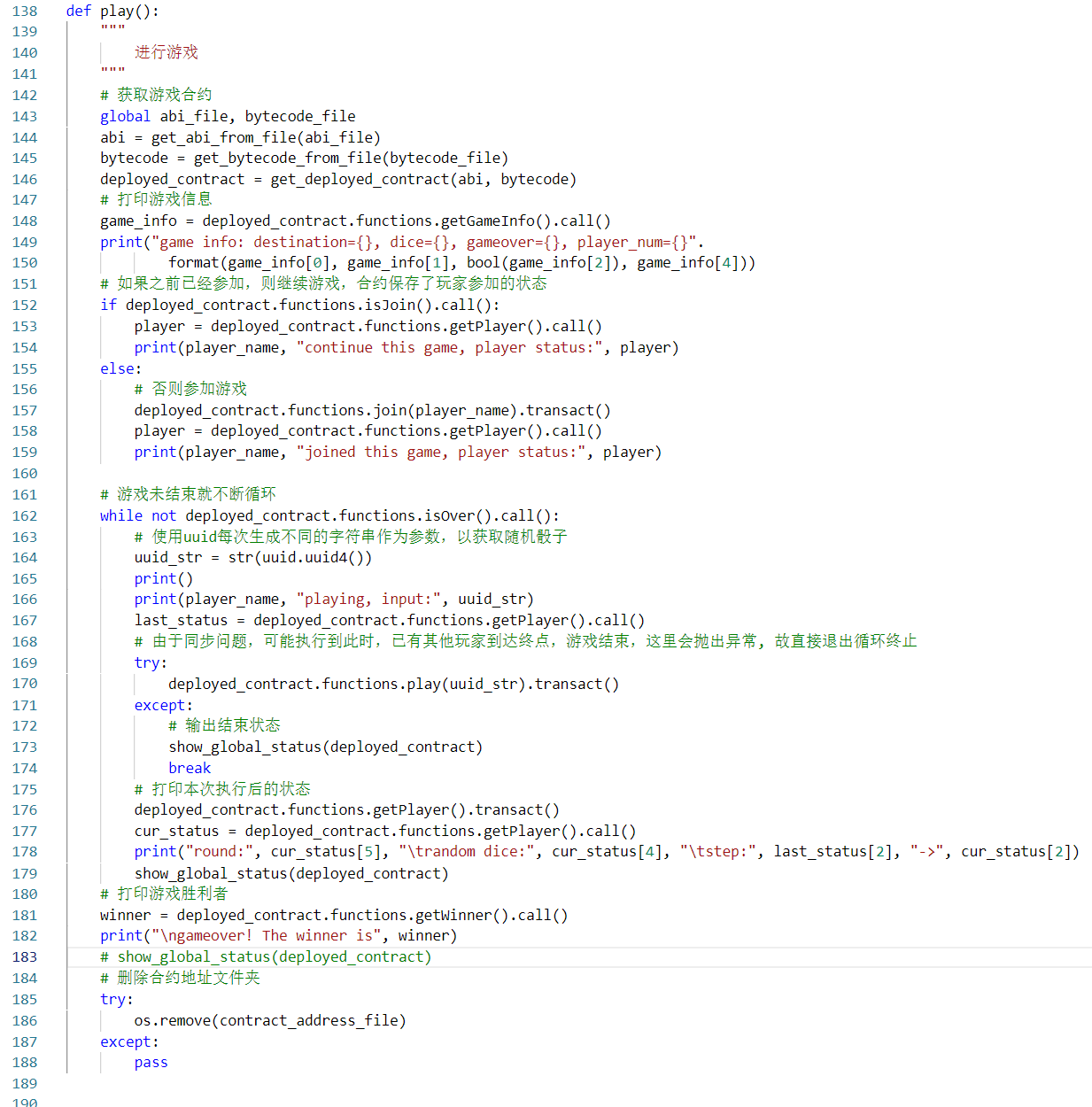


图 5-15 play函数

程序入口，先初始化玩家名称和账户id，然后设置默认账户，最后参与游戏：

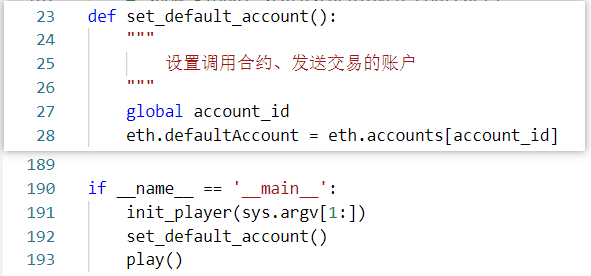


图 5-16 set\_default\_account函数

**【实验过程】**

如下图所示，写好合约后，使用solc进行编译或在remix上进行编译，其中- -overwrite是覆盖旧有的编译结果，便于修改后重新编译。然后就可以运行python代码，注意命令是python3，-n后面是玩家姓名（可自定），-i后面是以太坊账户eth.accounts数组的下标，所以不要超过gananche中账户的数目减一，同时相同的下标会被视作统一账户继续参赛，故开启多个终端时最好设定不同的姓名和下标值。最后一行python3命令可执行在多个终端中让多个玩家参与游戏合约。

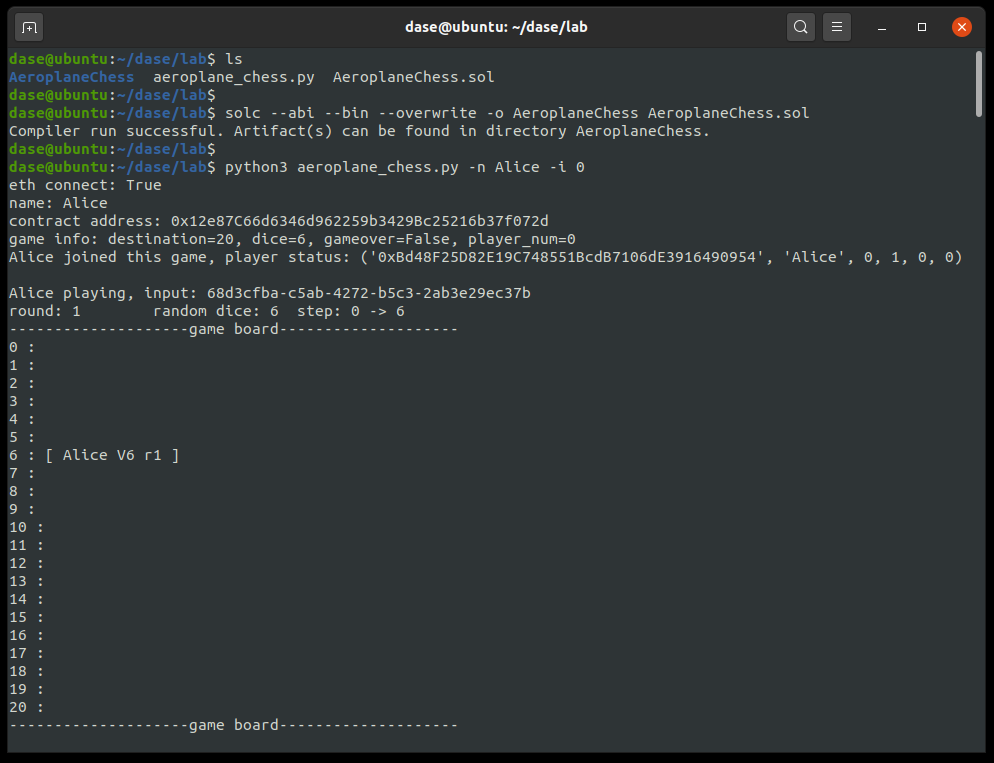


图 5-20 执行状态图

14轮后Alice到达终点，游戏终止。

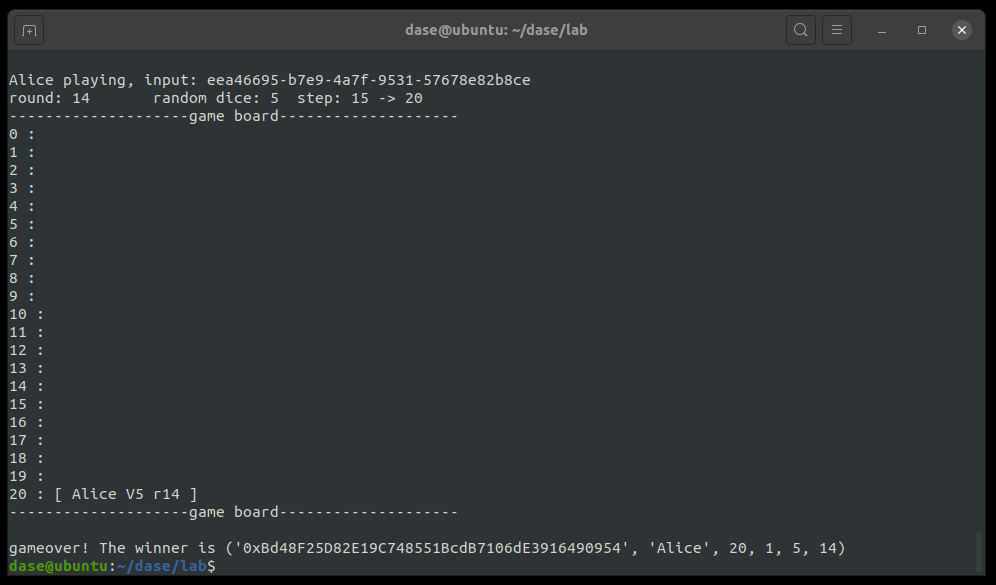


图 5-21 游戏结果图

多用户参加的示例在前文已给出，这里就不再演示。

**重新部署合约**

修改合约内容后，需要重新编译合约，删除文件目录中有“contract\_address.txt” 文件，重新运行python代码部署合约。如果不删除 “contract\_address.txt” 文件合约的修改将不能生效。

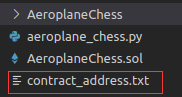


图 5-22 contract\_address.txt位置

**【实验小结】**

本实验通过一个有趣的飞行棋实验，介绍了Solidity的基本语法结构、使用Solidity编写智能合约的方法以及使用Python中的rpc部署和调用合约的过程，同时，简要介绍了以太坊可视化工具Ganache的使用方法。通过本实验的练习，读者将能够独立编写智能合约，能够用Python的rpc接口部署和调用智能合约。

读者可以在完成实验的基础上阅读更多的智能合约，了解在实际应用中智能合约的功能和结构，并尝试模仿其功能自己编写智能合约，加深对智能合约的理解。

**【习题】**

1. 在python代码中，确定调用哪个合约需要合约的哪些信息？

2. 想要快速地根据用户地址获取该用户的账户余额（整数），可以使用solidity编程中的什么数据结构？试定义一个具有上述功能的公共变量。

3. 在【实验过程】重新部署合约部分，需要删除“contract\_address.txt”文件的原因是什么？（在python代码中找答案）

**【参考文献】**

[1] <http://trufflesuite.com/ganache/>（Ganache）

[2] <https://solidity-cn.readthedocs.io/zh/develop/>（Solidity）

[3] <https://web3py.readthedocs.io/en/latest/>（Python Web3）

[4] （希）安德烈亚斯·M.安东波罗斯（Andreas M.Antonopoulos），（英）加文·伍德（Gavin Wood）著.精通以太坊 开发智能合约和去中心化应用[M].北京：机械工业出版社.2019.