一、代码及注释

```
bool Pig::useK()
{
    // TODO: 补全代码
    // 只能对攻击范围内的人使用(下家)
    // 攻击之前对敌我身份进行斟酌
    Pig *nxt = getNextPig();            // 攻击对象
    bool used = false;                  // 是否出杀

    // 主猪或者忠猪对反猪出杀
    if (this->type == 'M' || this->type == 'Z') {
        if (nxt->jumpType == 'F') {
            used = true;
        }
        // 主猪对类反猪出杀
        if (this->type == 'M') {
            if (nxt->jumpType == 'f') {
                used = true;
            }
        }
    }
    else {
        if (nxt->jumpType == 'Z') {
            used = true;
        }
    }
    if (used) {
        // 出杀表敌意肯定跳身份了
        this->jump();
        if (!nxt->del('D')) {
            nxt->hurt(this);
        }
        return true;
    }
    return false;
}

bool Pig::useF()
{
    // TODO: 补全代码
    // 忠猪会让着主猪
    // 攻击距离无限
    Pig *nxt;           // 攻击对象
    bool used = false;  // 是否打出该决斗

    // 确定决斗对象
    if (this->type == 'M' || this->type == 'Z') {
        nxt = getNextPig();
        do
        {
            if (nxt->jumpType == 'F') {
```

```
            used = true;
            break;
        }
        nxt = nxt->getNextPig();
    } while (nxt != this);

    if (!used) {
        nxt = getNextPig();
        if (this->type == 'M') {
            do
            {
                if (nxt->jumpType == 'f') {
                    used = true;
                    break;
                }
                nxt = nxt->getNextPig();
            } while (nxt != this);
        }
    }
}
else {
    nxt = getNextPig();
    do
    {
        if (nxt->type == 'M') {     // 第1个跳Z阵营的为MP
            used = true;
            break;
        }
        nxt = nxt->getNextPig();
    } while (nxt != this);
    // 反猪F只向主猪使用
    /*if (!used) {
        nxt = getNextPig();
        do
        {
            if (nxt->jumpType == 'Z') {
                used = true;
                break;
            }
            nxt = nxt->getNextPig();
        } while (nxt != this);
    }*/
}
if (used) {
    this->jump();

    // 找人帮忙
    if (nxt->findJ(this)) {
        return true;    // 被无懈了
    }
    /*if (this->findJ(nxt)) {
```

```
            return true;    // 被无懈了
        }*/

        if (this->type == 'M' && nxt->type == 'Z') {    // Z -> f,此时nxt未跳阵营
            nxt->hurt(this);
            return true;
        }

        // 轮流出杀
        while (true) {
            if (!nxt->del('K')) {
                nxt->hurt(this);
                break;
            }
            else if (this->type == 'M') {
                // 对方出杀，跳反
                nxt->jump();
            }
            if (!this->del('K')) {
                this->hurt(nxt);
                break;
            }
        }
        return true;
    }
    return false;
}

bool Pig::useN()
{
    for (Pig *nxt = getNextPig(); nxt != this; nxt = nxt->getNextPig())
    {
        // TODO: 补全代码
        if (nxt->findJ(this)) {
            continue;   // 被无懈了
        }
        if (!nxt->del('K')) {   // 出牌阶段除决斗外不会弃牌
            nxt->hurt(this);
            // 类反猪认定
            if (nxt->type == 'M') {
                if (this->jumpType == 0) {
                    this->jumpType = 'f';
                }
            }
        }
    }
    return true;
}

bool Pig::useW()
{
```

```
    for (Pig *nxt = getNextPig(); nxt != this; nxt = nxt->getNextPig())
    {
        // TODO: 补全代码
        // 无懈在锦囊生效之前使用
        if (nxt->findJ(this)) {
            continue;   // 被无懈了
        }
        if (!nxt->del('D')) {
            nxt->hurt(this);
            // 类反猪认定
            if (nxt->type == 'M') {
                if (this->jumpType == 0) {
                    this->jumpType = 'f';
                }
            }
        }
    }
    return true;
}

bool Pig::del(char c)
{
    // TODO: 补全代码
    for (auto it = cards.begin(); it != cards.end(); it++) {
        if (c == *it) {
            cards.erase(it);
            return true;
        }
    }
    return false;
}
```