跳棋游戏技术报告文档—宋建昊 2022010853

开发人员

- Name:宋建昊 Jianhao Song
- Student ID:2022010853
- Tel:15710565260
- Email:songjh22@mails.tsinghua.edu.cn

开发环境

- OS:Windows 11 64位操作系统
- Language:C++
- IDE:QT Creator

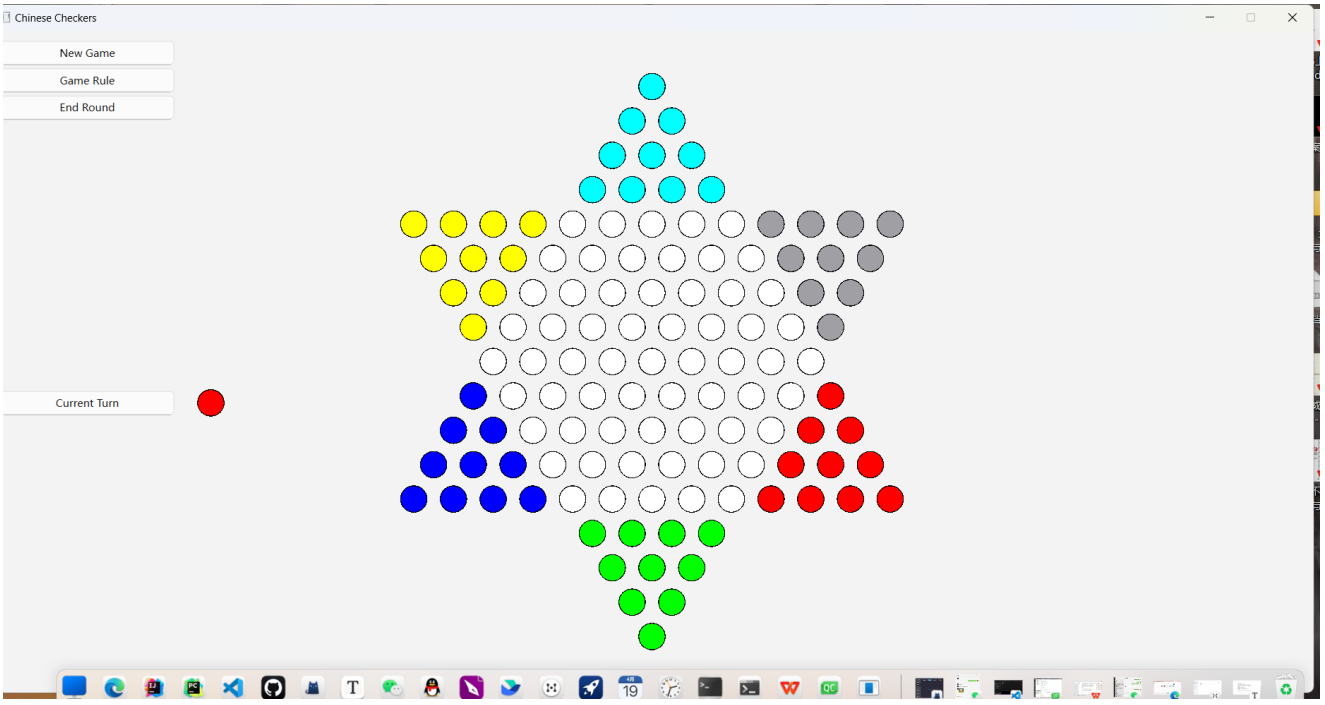
项目简介

本项目是基于面向对象程序设计思想，使用C++语言的QT框架，实现一个面向图形化界面的可供多玩家交互游戏的中国跳棋游戏。

游戏规则

- <游戏目标>
- 玩家轮流移动棋子，将自己颜色的所有棋子从起始角移动到正对面的角，最先完成这个目标的玩家获胜。
- <游戏人数>
- 支持2~6人同时游戏。玩家可以选择自己喜欢颜色的棋子，之后只需在自己选择的颜色的回合移动棋子，没有被选择的棋子颜色的回合直接跳过即可。
- <走棋规则>
- 每回合每位玩家只能移动至多一个棋子。
- 移动:棋子可以沿着与其相邻的六个方向中的任意一个方向移动一格到相邻的空格。
- 跳跃:如果相邻的位置上有任何颜色的棋子，且该棋子直线方向的下一个位置为空，则可以跳到该空位上.

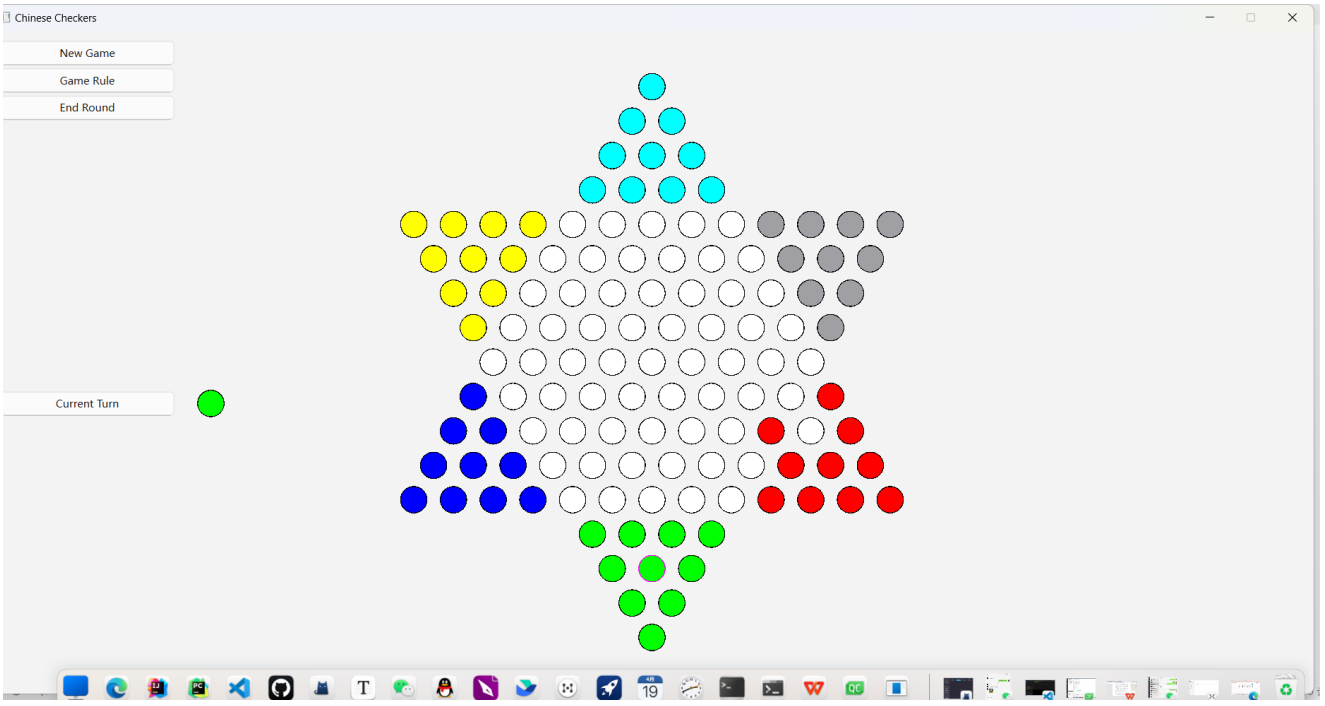
- 连续跳跃:在一次跳跃后，如果当前依然满足跳跃条件，可以继续跳跃，跳跃次数没有限制。
- <按钮说明>
- New Game:重新的一局，会更新棋盘。
- Game Rule:打开游戏规则页面，即当前的界面。
- End Round:结束这一回合，在一名玩家完成一回合的棋子移动后点击。
- <界面说明>
- 游戏界面[Current Turn]处会显示当前是哪位玩家的回合，在一名玩家点击[End Round]后会转移到下一名玩家的回合。
- 只能在界面上选中当前回合对应的玩家的棋子，选中的棋子会被红色边框圈出。
- <道德风尚>
- 游戏过程中，请不要对其他玩家恶语相向。
- 请不要因为输掉游戏而毁坏设备，或者殴打其他玩家。



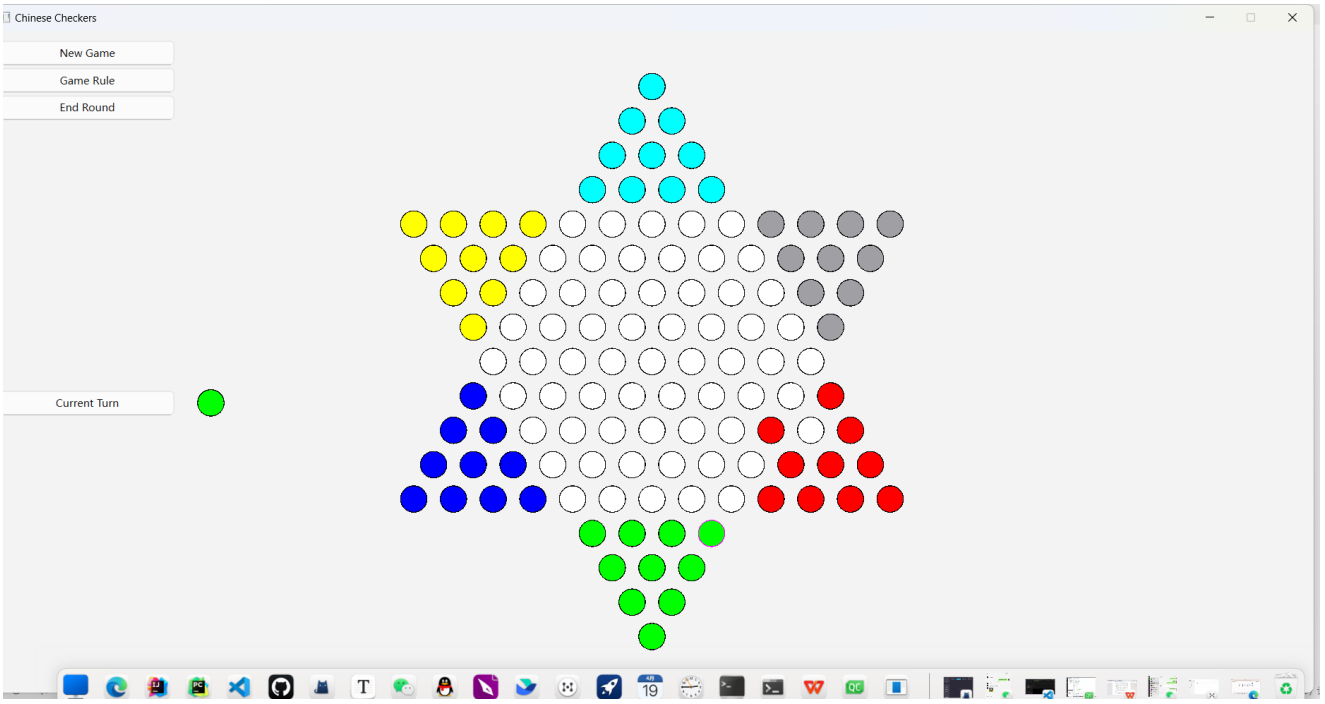
交互说明（运行结果）

基于上面的按钮说明与界面信息，接下来介绍游戏的交互说明。

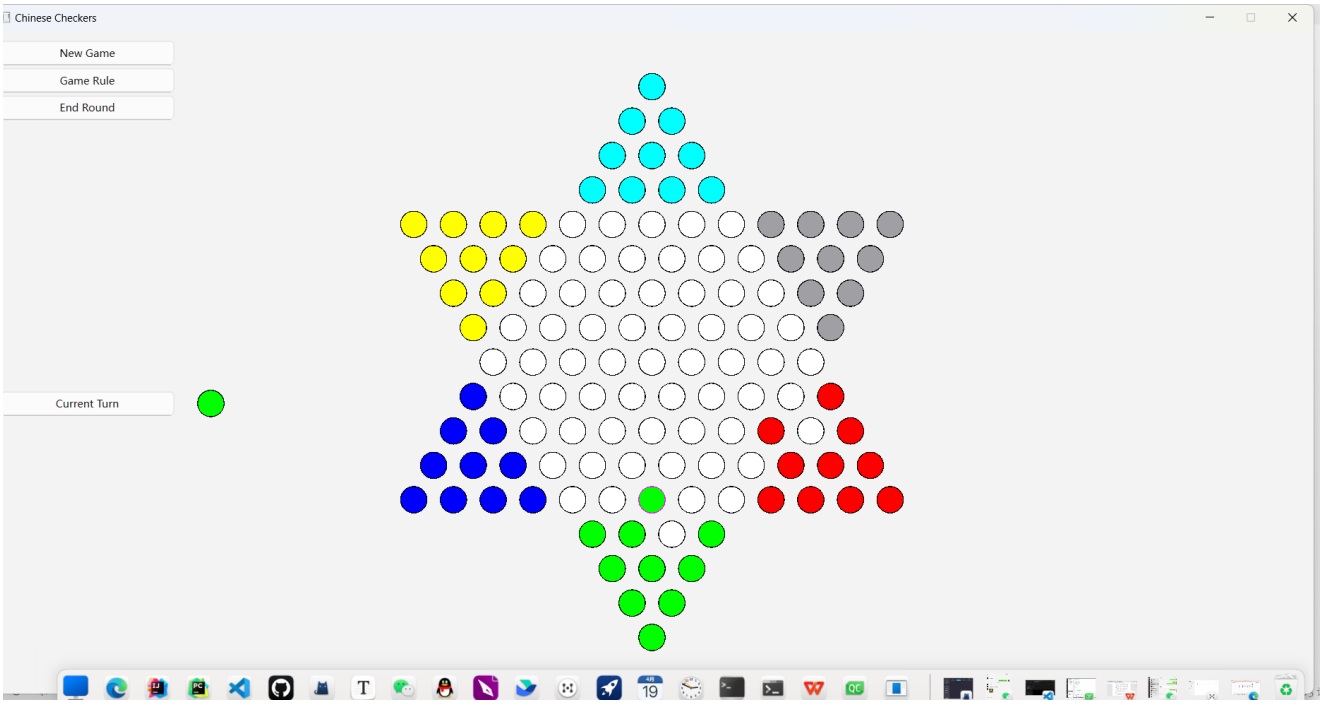
首先，玩家可以注意[Current Turn]处的按钮颜色，代表着当前回合的玩家颜色，只有当前回合对应的棋子可以被选中，例如现在是绿玩家的回合，只能成功选中绿色棋子，点击其他颜色的棋子不会被选中，选中棋子会被红圈框住，如下图。



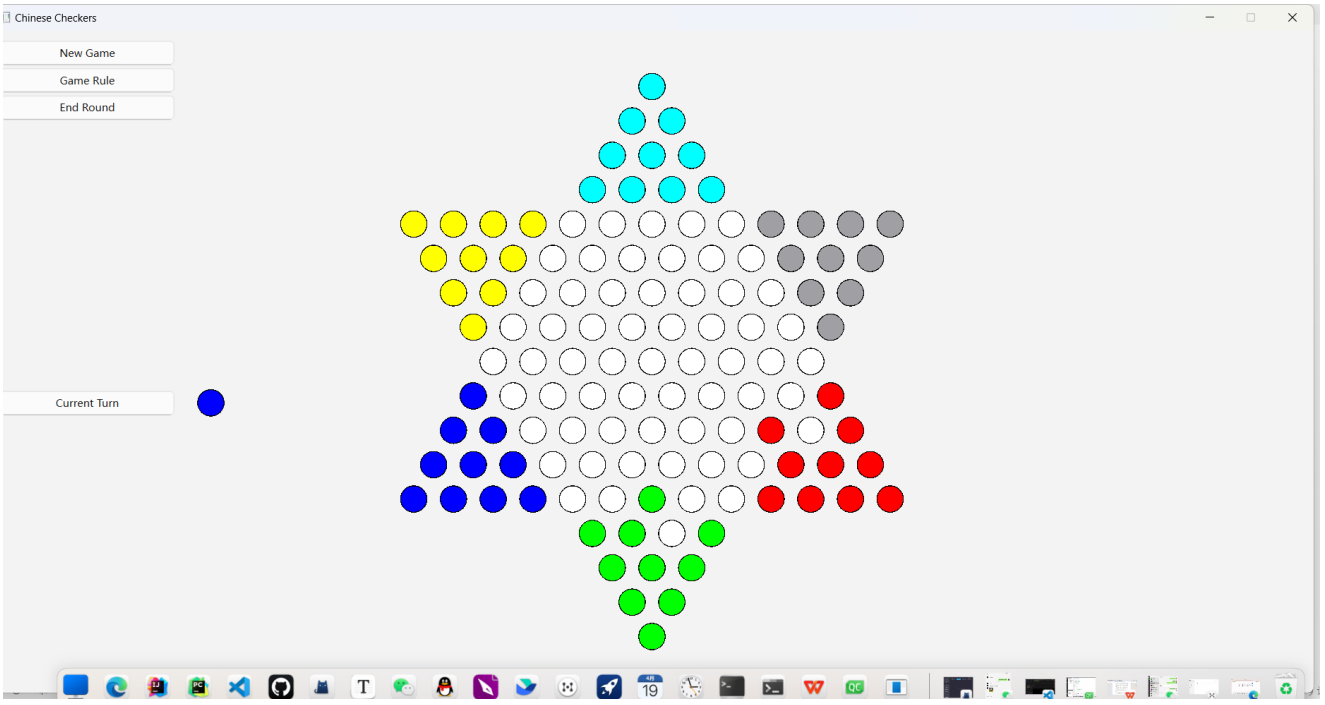
基于跳棋规则，每回合最多只能移动一个棋子，所以在这一回合已经进行过棋子移动之后就不可以更改选中的棋子了，只有在尚未移动棋子时可以更改选中的棋子，比如现在绿色玩家还没移动棋子，临时决定移动另一个棋子，那么他直接点击另一个绿色棋子就可以更改选中，但如果在移动之后再点击其他绿色棋子的操作不会被响应。下图为成功更改选中棋子后。



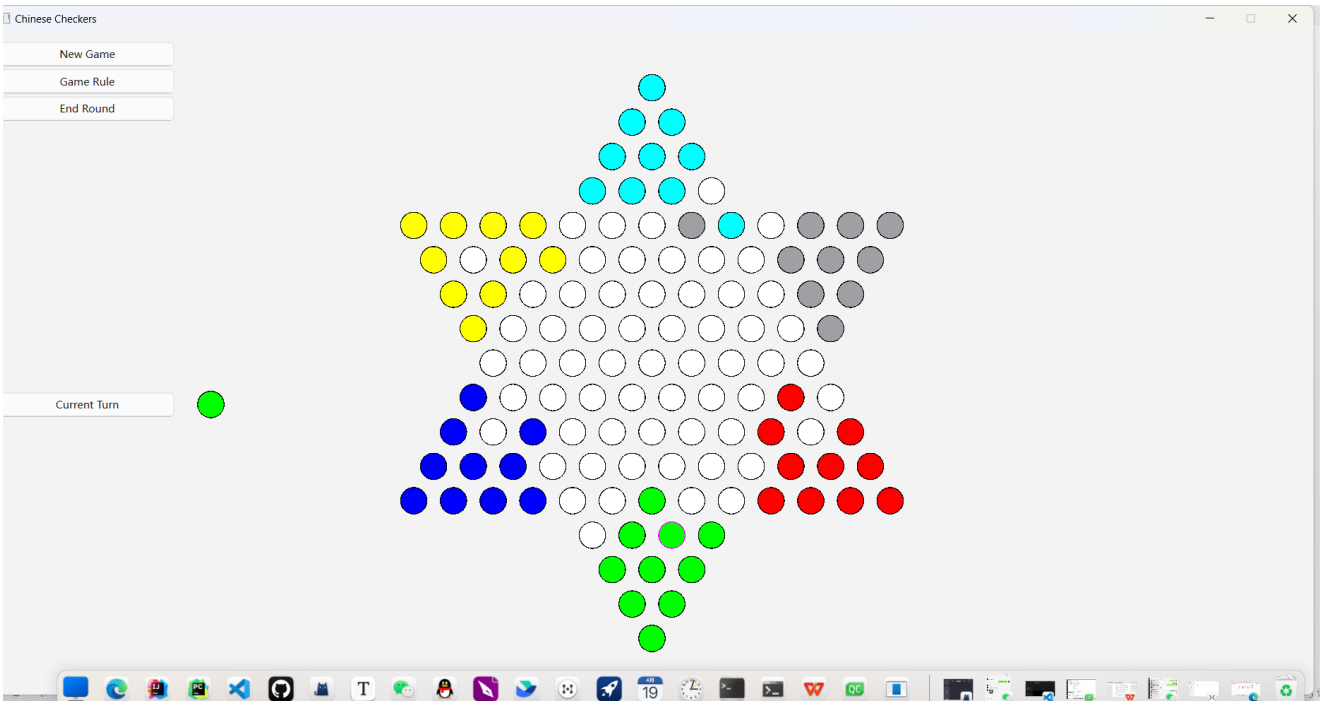
移动棋子的操作也很简单，只需要选中棋子之后点击目标位置就可以了，程序会自动检验移动操作是否符合跳棋规则，如何点击了不合法的目标位置则不会成功移动棋子，点击合法的目标位置之后棋子会被成功移动，而选中棋子仍然是这个被移动的棋子不会被改变，这时这回合用户就只能这一个棋子了。下图为成功移动棋子后。

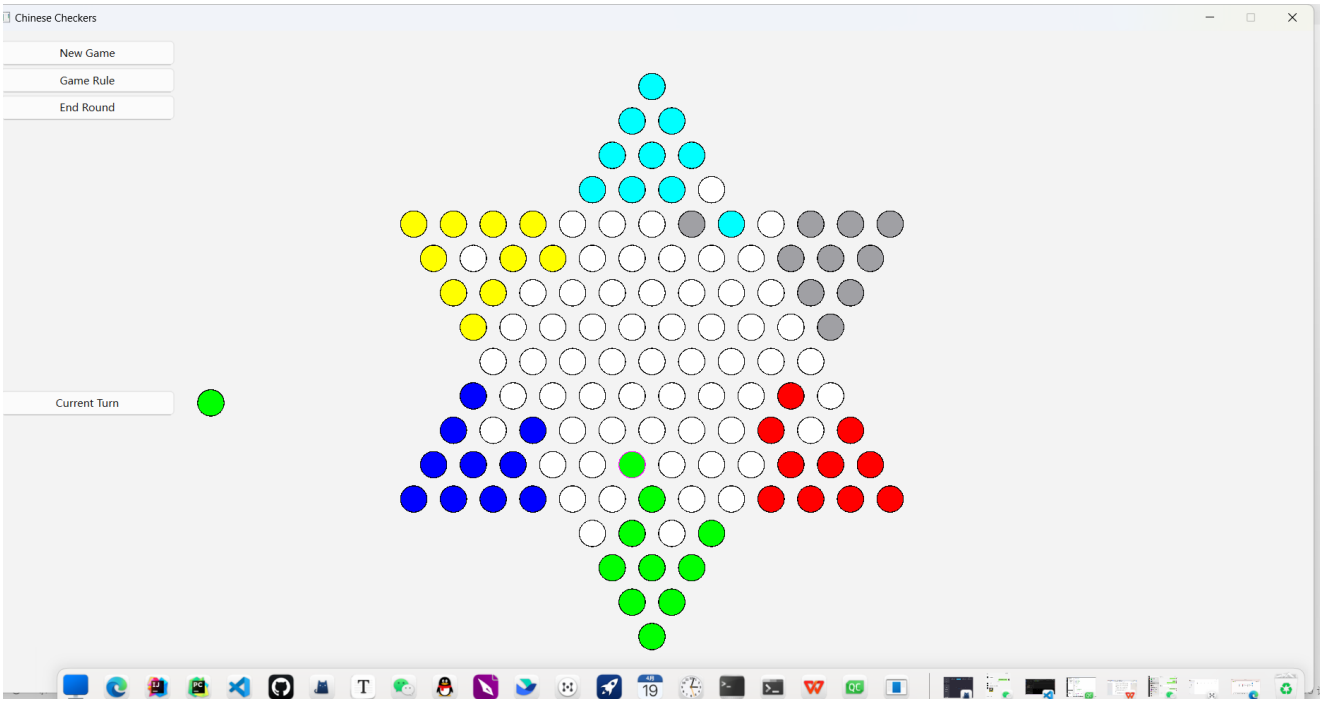


玩家结束当前回合的操作（可以选择不进行移动，直接跳过本回合）后，只需点击[End Round]按钮就可以结束本回合，这时按顺时针顺序轮到下一位玩家的回合，[Current Turn]处会更新当前回合棋子颜色方便玩家参考。这时上一回合的选中棋子会被取消，需要重新选择选中棋子。这位玩家完成操作后再次点击[End Round]结束回合，以此类推，玩家按规则顺时针循环依次操作。下图为点击按钮[End Round]结束回合后，此时轮到下一位玩家的回合。

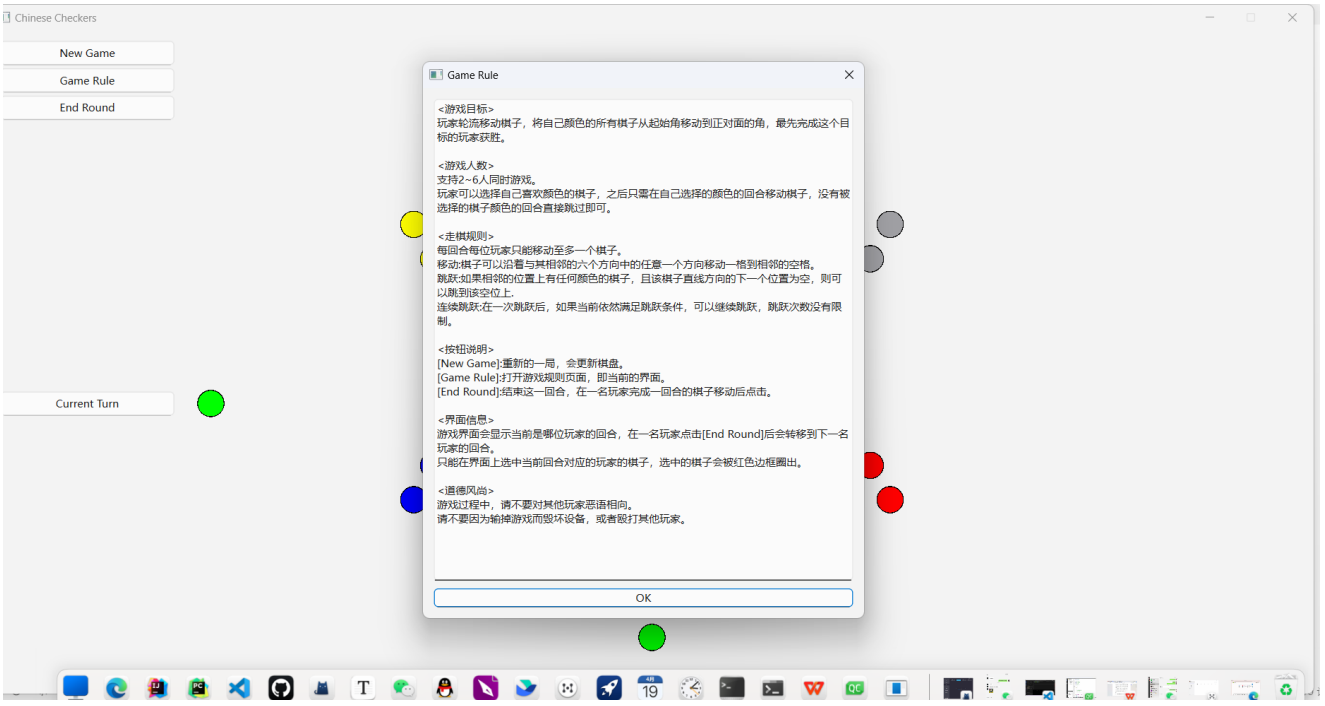


注意移动规则中，除了移动到相邻的空位置和隔着一个相邻的棋子跳跃到该方向下一个空位置之外，玩家还可以进行连跳操作。注意只有在玩家在本回合只进行了跳跃操作时才可以进行连跳，而且进行过跳跃之后不能再移动到相邻的空位置。游戏程序会自动判断移动是否符合规则，如果本次移动不合法则棋子不会被移动，连跳时只需要依次点击路径上的目标位置即可，下面两图为此时最左侧的绿色棋子进行连跳。

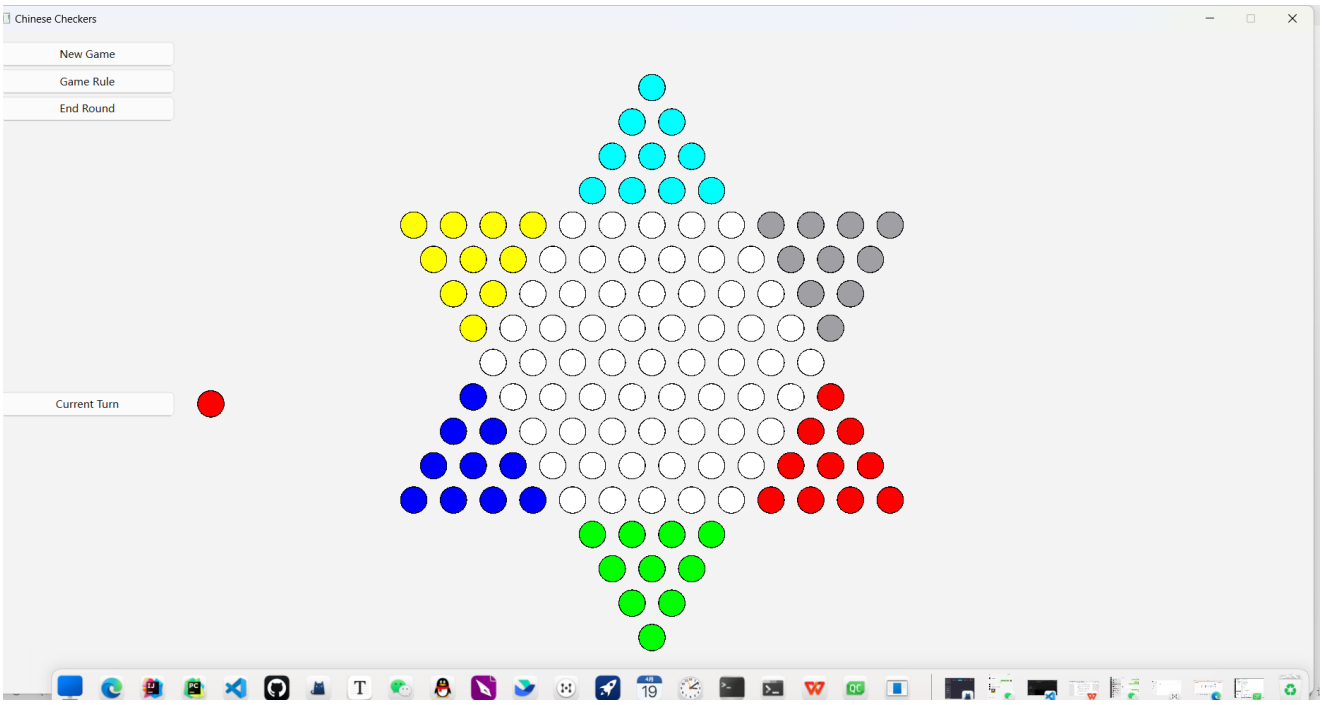




在游戏中途，如果玩家希望可以参考游戏规则和操作说明，可以点击[Game Rule]按钮，点击之后会跳出内容为由游戏规则和操作说明的界面，此界面为不可编辑状态，点击界面上的[OK]即可返回游戏。下图为点击后跳出界面。



如果玩家希望结束这一句游戏并重新开一局游戏只需点击[New Game]按钮即可，所有棋子会被放回原位。效果如下。



代码实现

主类（Checherboard）及其构造函数

```
class Checkerboard : public QWidget {
public:
    Checkerboard(QWidget *parent = nullptr) : QWidget(parent) {
        // 设置窗口标题与大小
        setWindowTitle(tr("Chinese Checkers"));
        boardSize = 750;
```

```
        setFixedSize(2*boardSize, boardSize);

        //设置state[i][j][k]位点状态数组
        //0~5分别代表六种颜色的棋子，-1代表空位点没有棋子
        //...

        //设置按钮大小与界面布局
        QPushButton *refleshButton = new QPushButton("New Game", this);
        QPushButton *ruleButton = new QPushButton("Game Rule", this);
        QPushButton *nextButton = new QPushButton("End Round", this);
        QPushButton *turnButton = new QPushButton("Current Turn", this);

        refleshButton->setSizePolicy(QSizePolicy::Fixed,
QSizePolicy::Fixed);
        ruleButton->setSizePolicy(QSizePolicy::Fixed,
QSizePolicy::Fixed);
        nextButton->setSizePolicy(QSizePolicy::Fixed,
QSizePolicy::Fixed);
        turnButton->setSizePolicy(QSizePolicy::Fixed,
QSizePolicy::Fixed);

        refleshButton->setFixedSize(200, 30);
        ruleButton->setFixedSize(200, 30);
        nextButton->setFixedSize(200, 30);
        turnButton->setFixedSize(200, 30);

        QVBoxLayout *layout = new QVBoxLayout(this);
        layout->setSpacing(1);
        layout->setContentsMargins(10, 10, 10, 10);

        layout->addWidget(refleshButton);
        layout->addWidget(ruleButton);
        layout->addWidget(nextButton);
        layout->addStretch();
        layout->addWidget(turnButton);
        layout->addStretch();

        //使用槽函数连接按钮点击操作与对应的执行函数
        connect(refleshButton, &QPushButton::clicked, this,
&Checkerboard::reflesh);
        connect(nextButton, &QPushButton::clicked, this,
&Checkerboard::nextround);
        connect(ruleButton,
&QPushButton::clicked,this,&Checkerboard::showrule);

        //初始化游戏参数
        turn=0;
        centrestate=-1;
        present=coord(6,6,6);
        jump=false;
```

```
        movement=false;

    }

    //成员变量
    private:
        int boardSize; // 棋盘大小

        coord present;//当前棋子坐标
        int turn;//当前回合
        bool jump;//当前回合是否进行了跳跃
        bool movement;//当前回合是否移动了棋子

        QPointF centre;//中心位点坐标
        QPointF position[6][4][5];//棋盘位置坐标
        int centrestate;//中心位点状态
        int state[6][4][5];//棋盘位点状态
}
```

重写绘制函数

```
protected:
    void paintEvent(QPaintEvent *event) override {
        Q_UNUSED(event);
        //设置画笔
        QPainter painter(this);
        // 绘制棋盘界面
        drawCheckerboard(painter);

    }

    void drawCheckerboard(QPainter &painter) {
        // 计算棋盘中心的顶点坐标与位点间距
        double centerX = boardSize;
        double centerY = boardSize / 2;
        double radius = 3*boardSize / 50;

        //根据游戏参数设置边框颜色与填充颜色
        //边框颜色决定是否选中效果，填充颜色决定棋子种类
        //painter.setPen(...)
        //painter.setBrush(...)

        //绘制本回合棋子
        QPointF turing = QPointF(centerX/3,9*centerY/8);
        painter.drawEllipse(turing, radius/3, radius/3);


        //根据游戏参数设置边框颜色与填充颜色
```

```
        //边框颜色决定是否选中效果，填充颜色决定棋子种类
        //painter.setPen(...)
        //painter.setBrush(...)

        //绘制中心棋子
        centre = QPointF(centerX,centerY);
        painter.drawEllipse(centre, radius/3, radius/3);

        //三层循环为棋子位点的数组坐标
        for (int i = 0; i < 6; ++i) {
            double angle1 = M_PI / 3 * i;
            double angle2 = M_PI / 3 * (i+1);
            for (int j = 0; j < 4; j++ ){
                for(int k = 0; k < 5; k++){

                    //确定每个位点的二维坐标
                    double x=centerX+
(j+1)*radius*qCos(angle1)+k*radius*qCos(angle2);
                    double y=centerY+
(j+1)*radius*qSin(angle1)+k*radius*qSin(angle2);

                    //根据游戏参数设置边框颜色与填充颜色
                    //边框颜色决定是否选中效果，填充颜色决定棋子种类
                    //painter.setPen(...)
                    //painter.setBrush(...)

                    //绘制普通点位
                    position[i][j][k]= QPointF(x,y);
                    painter.drawEllipse(position[i][j][k], radius/3,
radius/3);
                }

            }

        }
    }
```

一个数组三维坐标类与三个工具函数

```
//三维数组坐标类
class coord {
public:
    int r;
    int s;
    int t;

    // Default constructor
    coord() : r(0), s(0), t(0) {}
    // Parameterized constructor
```

```
    coord(int rValue, int sValue, int tValue) : r(rValue), s(sValue),
t(tValue) {}

    // 重载判等运算符，只有在三个坐标都相等时判定为相等
    bool operator==(const coord& other) const {
        return r == other.r && s == other.s && t == other.t;
    }
};

//通过二位坐标获得数组坐标
coord fetch(QPointF point){

        double centerX = boardSize;
        double centerY = boardSize / 2;
        double radius = 3 * boardSize / 50;

        //常规点判定
        for (int i = 0; i < 6; ++i) {

            double angle1 = M_PI / 3 * i;
            double angle2 = M_PI / 3 * (i+1);

            for (int j = 0; j < 4; j++ ){
                for(int k = 0; k < 5; k++){

                    double x=centerX+
(j+1)*radius*qCos(angle1)+k*radius*qCos(angle2);
                    double y=centerY+
(j+1)*radius*qSin(angle1)+k*radius*qSin(angle2);
                    position[i][j][k]= QPointF(x,y);

                    //如果曼哈顿距离足够小，则判定为对应的点
                    if ((QPointF(x,y) - point).manhattanLength() < 30) {
                        return coord(i,j,k);
                        break;
                    }


                }

            }
        }
        //中心点判定
        if ((QPointF(centerX,centerY) - point).manhattanLength() < 30) {
            return coord(-1,-1,-1);
        }
        //无效点判定
        return coord(6,6,6);
    }
```

```
    bool isValidMove(coord present, coord now)  {
        //选接两个参数：选中位点与目标位点
       //判定是否为合法的有效移动
    }

    void mousePressEvent(QMouseEvent* event) override
    {
        //根据当前游戏参数，以及参数：鼠标点击事件的位点位置
        //利用上面两个工具函数，判定当前发生的操作类型：无效操作/更改选中/移动棋子/...
        //如果需要的话，更改游戏参数

        //刷新页面
        update();
        return;
    }
```

更新按钮与结束回合按钮

```
    //槽函数连接按钮与对应函数
 connect(refleshButton, &QPushButton::clicked, this, &Checkerboard::reflesh);
 connect(nextButton, &QPushButton::clicked, this, &Checkerboard::nextround);

//更新函数，修改函数参数将棋子与状态归位，
void reflesh(){
        for (int i = 0; i < 6; ++i) {
            for (int j = 0; j < 4; j++ ){
                for(int k = 0; k < 5; k++){

                    if( j+k >= 4){
                        state[i][j][k]=i;
                    }else{
                        state[i][j][k]=-1;
                    }

                }

            }
        }
        turn=0;
        centrestate=-1;
        present=coord(6,6,6);
        jump=false;
        movement=false;
```

```
        update();
        return;
    }

//结束当前回合，修改回合参数以及其他有必要修改的参数
    void nextround(){
        turn++;
        if(turn==6){
            turn=0;
        }
        present=coord(6,6,6);
        jump=false;
        movement=false;
        update();
        return;
    }
```

规则说明按钮及其实现

```
//规则提示窗口类
class rulesDialog : public QDialog
{
    Q_OBJECT

public:
    rulesDialog(QWidget* parent = nullptr) : QDialog(parent)
    {

        //窗口题目
        this->setWindowTitle("Game Rule");
        // 设置窗口为模态，防止用户点击其他窗口
        this->setModal(true);
        QVBoxLayout *layout = new QVBoxLayout(this);

        //初始化文本编辑栏
        QTextEdit *rulesTextEdit = new QTextEdit(this);

        //设置规则提示的文本内容
```

```
        rulesTextEdit->setPlainText("<游戏目标>\n玩家轮流移动棋子，将自己颜色的所有棋子从起始角移动到正对面的角，最先完成这个目标的玩家获胜。\n\n<游戏人数>\n支持2~6人同时游戏。\n玩家可以选择自己喜欢颜色的棋子，之后只需在自己选择的颜色的回合移动棋子，没有被选择的棋子颜色的回合直接跳过即可。\n\n<走棋规则>\n每回合每位玩家只能移动至多一个棋子。\n移动:棋子可以沿着与其相邻的六个方向中的任意一个方向移动一格到相邻的空格。\n跳跃:如果相邻的位置上有任何颜色的棋子，且该棋子直线方向的下一个位置为空，则可以跳到该空位上.\n连续跳跃:在一次跳跃后，如果当前依然满足跳跃条件，可以继续跳跃，跳跃次数没有限制。\n\n<按钮说明>\n[New Game]:重新的一局，会更新棋盘。\n[Game Rule]:打开游戏规则页面，即当前的界面。\n[End Round]:结束这一回合，在一名玩家完成一回合的棋子移动后点击。\n\n<界面信息>\n游戏界面会显示当前是哪位玩家的回合，在一名玩家点击[End Round]后会转移到下一名玩家的回合。\n只能在界面上选中当前回合对应的玩家的棋子，选中的棋子会被红色边框圈出。\n\n<道德风尚>\n游戏过程中，请不要对其他玩家恶语相向。\n请不要因为输掉游戏而毁坏设备，或者殴打其他玩家。\n");

        // 设置为只读
        rulesTextEdit->setReadOnly(true);
        rulesTextEdit->setFocusPolicy(Qt::NoFocus);
        layout->addWidget(rulesTextEdit); // 将文本编辑框添加到布局中
        QPushButton *okButton = new QPushButton("OK", this);
        //槽函数连接[OK]按钮与完毕信号
        connect(okButton, &QPushButton::clicked, this, &QDialog::accept);
        layout->addWidget(okButton); // 将OK按钮添加到布局中

        // 设置布局到对话框
        this->setLayout(layout);

        // 设置对话框的初始大小
        this->resize(500, 600); // 您可以根据需要调整这个大小

    }
};

   //初始化规则提示界面，收到完毕信号状态就返回游戏
    void showrule(){
        rulesDialog dialog(this);
        if(dialog.exec()== QDialog::Accepted){
            return;
        }
    }
```

主函数

```
#include "mainwindow.h"

#include <QApplication>

int main(int argc, char *argv[]) {
    QApplication app(argc, argv);
    Checkerboard checkerboard;
    checkerboard.show();
    return app.exec();
}
```