Android Assignment 1 – Ingress Chess

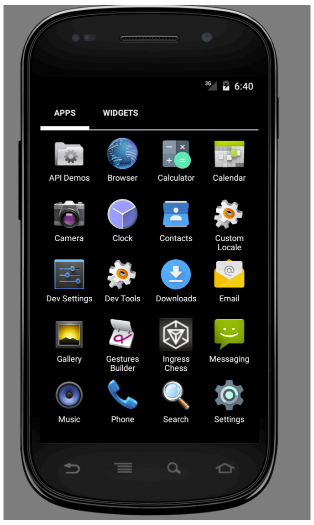
Name: Cao Siyuan

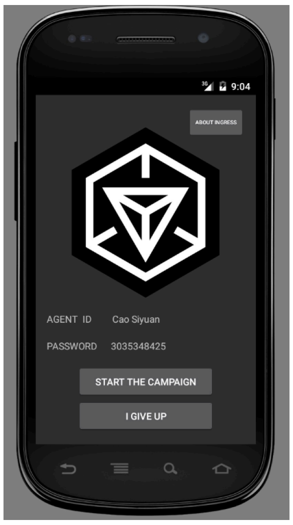
StudentID: 3035348425

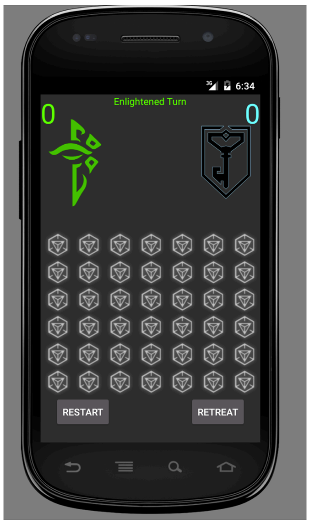
Development Environment

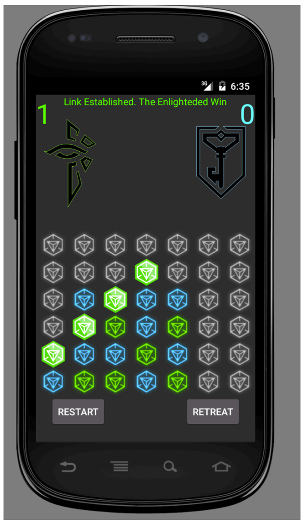
- OSX 10.11.6
- Android Studio 1.4.1
- Android 5.1.1 (API 22)
- AVD (Nexus S with resolution 800x480)

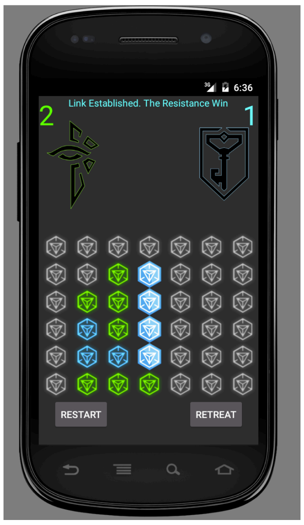
Screenshot

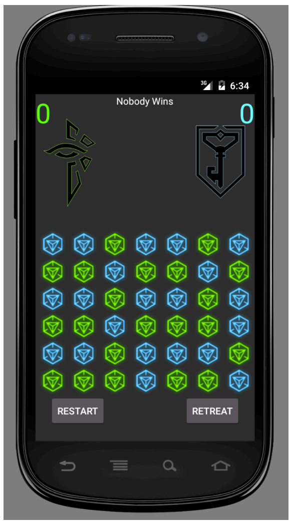


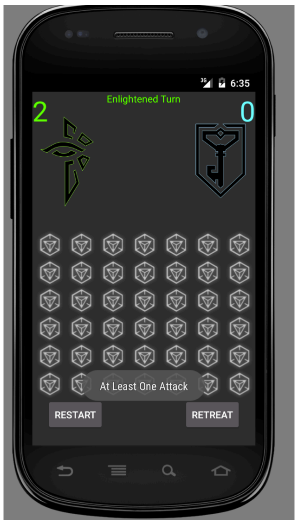




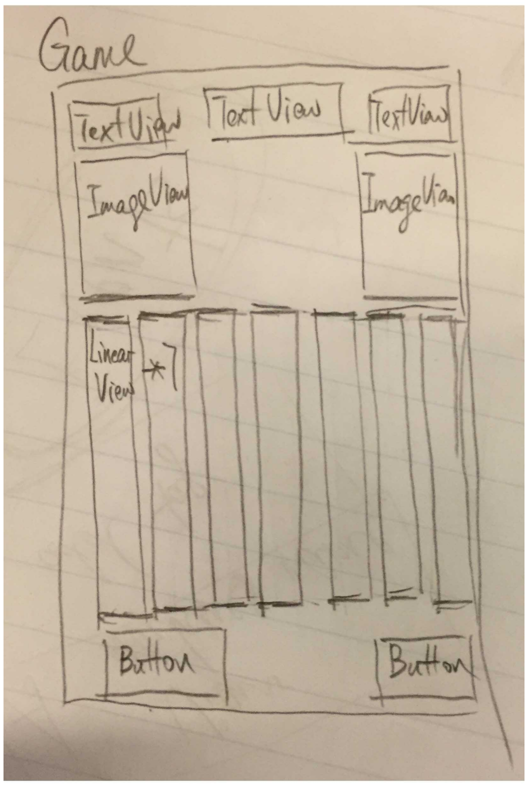


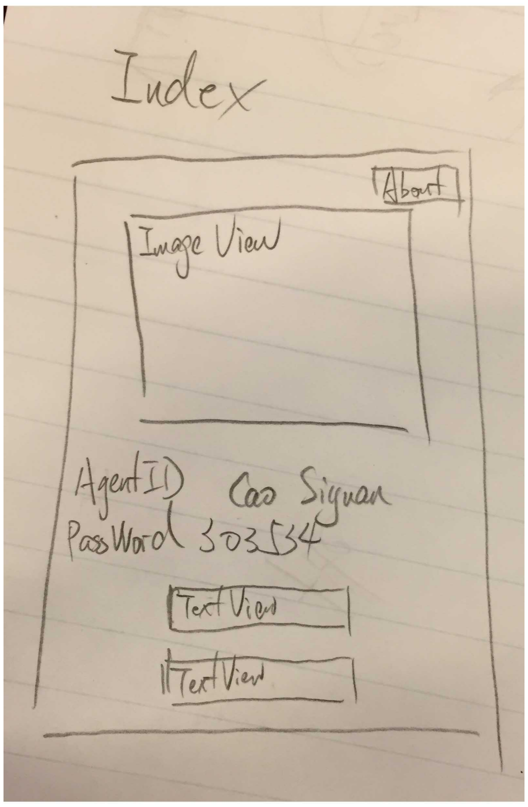






Design





Code

valuables design

```java
    private boolean isRedTurn;
    private int[][] chess = new int[7][6];
    private ArrayDeque<Integer> sequence = new ArrayDeque<>();
    private ImageView imageButton[][] = new ImageView[7][6];
    private boolean gameSet = false;
    private ImageView blueTurn, greenTurn;
    private int[] score = {0,0};
```

which

- `isRedTurn` is set to figure out whose turn
- `chess` is a two dimension Array to store how the chess table is like:
    - none = 0

    - red = 1
    - green =2
    - fin = 3
    - hired = 4
    - higreen = 5

- `sequence` is a stack to implement the **retreat** feature
- `gameSet` is set to figure out if someone wins
- `score` is the score of both two sides

Drop chess

```java
imageButton[i][j].setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        if(!gameSet) {
            if (!isRedTurn) {
                localgameTurn.setText("Resistance Turn");
                localgameTurn.setTextColor(Color.parseColor("#00FFFF"));
                localblueTurn.setImageResource(R.drawable.resistance_alt);
                localgreenTurn.setImageResource(R.drawable.enlightened);
            } else {
                localgameTurn.setText("Enlightened Turn");
                localgameTurn.setTextColor(Color.GREEN);
                localblueTurn.setImageResource(R.drawable.resistance);
                localgreenTurn.setImageResource(R.drawable.enlightened_alt);

            }
        }
        if (chess[locali][localj] == 0) {
            for (int count = 5; count >= 0; count--) {
                if (chess[locali][count] == 0 && !gameSet) {
                    if (isRedTurn) {
                        localimageButton[locali][count].setImageResource(R.drawable.
                        isRedTurn = false;
                        sequence.push(locali);
                        sequence.push(count);
                        chess[locali][count] = 1;
                        ifWin(locali, count, 1);
                        break;
                    } else {
                        localimageButton[locali][count].setImageResource(R.drawable.
                        isRedTurn = true;
                        sequence.push(locali);
                        sequence.push(count);
                        chess[locali][count] = 2;
                        ifWin(locali, count, 2);
                        break;
                    }
                }
            }
        }
    }
});
```

Judge Draw

Java
```
protected void drawGame(){
        if (sequence.size() == 84){
            gameSet = true;
            Toast.makeText(game.this, "Draw Game", Toast.LENGTH_SHORT).show();
            blueTurn.setImageResource(R.drawable.resistance);
            greenTurn.setImageResource(R.drawable.enlightened);
            gameTurn.setText("Nobody Wins");
            gameTurn.setTextColor(Color.parseColor("#FFFFFF"));

            gscore.setText(Integer.toString(score[1]));
            gscore.setTextColor(Color.GREEN);

            rscore.setText(Integer.toString(score[0]));
            rscore.setTextColor(Color.parseColor("#00FFFF"));
        }
    }
```

Judge Win

Java
```
protected void ifWin(int i, int j, int color) {
    horWin(i, j, color);
    verWin(i, j, color);
    lcrossWin(i, j, color);
    rcrossWin(i, j, color);
    winEffect(i, j, color);
    drawGame();
}
protected void horWin(int i, int j, int color) {
    //确保落下子的值还是原来的color
    chess[i][j] = color;
    int lcount = 0;
    int rcount = 0;
    //下面两个for判断横向4子
    for (int x = i; x >= 0; x--) {
        //判左边
        if (chess[x][j] == color) {
            lcount++;
        } else {
            break;
        }
    }
    for (int x = i; x < 7; x++) {
        //判右边
        if (chess[x][j] == color) {
            rcount++;
        } else {
            break;
        }
    }
    if (lcount + rcount >= 5) {
        //全部放到一个数组里
        for (int x = i - lcount + 1; x < i + rcount; x++) {
            if (color == 1) {
                chess[x][j] = 4;
            } else {
                chess[x][j] = 5;
            }
        }
        gameSet = true;
    }
}
protected void verWin(int i, int j, int color) {
    int dcount = 0;
    //确保落下子的值还是原来的color
    chess[i][j] = color;
    //下面一个for判断上下4子
    for (int x = j; x < 6; x++) {
        if (chess[i][x] == color) {
            dcount++;
        } else {
            break;
        }
    }
    if (dcount >= 4) {
        //放到一个数组里
        for (int x = j; x < j + dcount; x++) {
            if (color == 1) {
                chess[i][x] = 4;
            } else {
                chess[i][x] = 5;
            }
        }
        gameSet = true;
    }
}
protected void lcrossWin(int i, int j, int color) {
    //确保落下子的值还是原来的color
    chess[i][j] = color;
    int drcount = 0;
    int ulcount = 0;
    //下面2个for判断左对角线4子
    int tmpj = j;
    for (int x = i; x >= 0 && tmpj >= 0; x--, tmpj--) {
        //判左上
```

```
            if (chess[x][tmpj] == color) {
                ulcount++;
            } else {
                break;
            }
        }
        tmpj = j;
        for (int x = i; x < 6 && tmpj < 5; x++, tmpj++) {
            //判右下
            if (chess[x + 1][tmpj + 1] == color) {
                drcount++;
            } else {
                break;
            }
        }
        if (ulcount + drcount >= 4) {
            //全部放到一个数组里
            for (int hix = i - ulcount + 1, hij = j - ulcount + 1; hix < i + drcount + 1
                if (color == 1) {
                    chess[hix][hij] = 4;
                } else {
                    chess[hix][hij] = 5;
                }
            }
            gameSet = true;
        }
    }
    protected void rcrossWin(int i, int j, int color) {
        //确保落下子的值还是原来的color
        chess[i][j] = color;
        int urcount = 0;
        int dlcount = 0;
        //下面一个for判断右对角线4子
        int tmpj = j;
        for (int x = i; x < 7 && tmpj >= 0; x++, tmpj--) {
            //判右上
            if (chess[x][tmpj] == color) {
                urcount++;
            } else {
                break;
            }
        }
        tmpj = j;
        for (int x = i; x > 0 && tmpj < 5; x--, tmpj++) {
            //判左下
            if (chess[x - 1][tmpj + 1] == color) {
                dlcount++;
            } else {
```

```
            break;
        }
    }

    if (urcount + dlcount >= 4) {
        //全部放到一个数组里
        for (int hix = i - dlcount, hij = j + dlcount; hix < i + urcount; hix++, hij
            if (color == 1) {
                chess[hix][hij] = 4;
            } else {
                chess[hix][hij] = 5;
            }
        }
        gameSet = true;
    }
}
```

Restart

Java

```
restart.setOnClickListener(new View.OnClickListener(){
    @Override
    public void onClick(View v){
        //初始化chess数组
        for (int i = 0; i < 7; i++) {
            for (int j = 0; j < 6; j++) {
                chess[i][j] = 0;
                imageButton[i][j].setImageResource(R.drawable.empty_t);

            }
        }
        isRedTurn = false;
        gameSet = false;
        localgameTurn.setText("Enlightened Turn");
        localgameTurn.setTextColor(Color.GREEN);
        sequence.clear();
    }
});
```

Retreat

Java
```java
retreat.setOnClickListener(new View.OnClickListener(){
    @Override
    public void onClick(View v) {
        if(!gameSet){
            if(sequence.isEmpty()){
                Toast.makeText(game.this, "At Least One Attack", Toast.LENGTH_SHORT
            }else {
                isRedTurn = !isRedTurn;
                int retreatj = sequence.pop();
                int retreati = sequence.pop();
                chess[retreati][retreatj] = 0;
                imageButton[retreati][retreatj].setImageResource(R.drawable.empty_t

                if (isRedTurn) {
                    localgameTurn.setText("Resistance Turn");
                    localgameTurn.setTextColor(Color.parseColor("#00FFFF"));
                    localblueTurn.setImageResource(R.drawable.resistance_alt);
                    localgreenTurn.setImageResource(R.drawable.enlightened);
                } else {
                    localgameTurn.setText("Enlightened Turn");
                    localgameTurn.setTextColor(Color.GREEN);
                    localgreenTurn.setImageResource(R.drawable.enlightened_alt);
                    localblueTurn.setImageResource(R.drawable.resistance);
                }
            }
        }else{
            Toast.makeText(game.this, "Lose is Lose. One More Campaign", Toast.LENG
        }
    }
});
```

Limitation

Some animation may make this game better.

Reference

- Official API Reference
  - https://developer.android.com/index.html
- Ingress ICO
  - https://github.com/cr0ybot/ingress-logos
- COMP7506 Lecture Slides