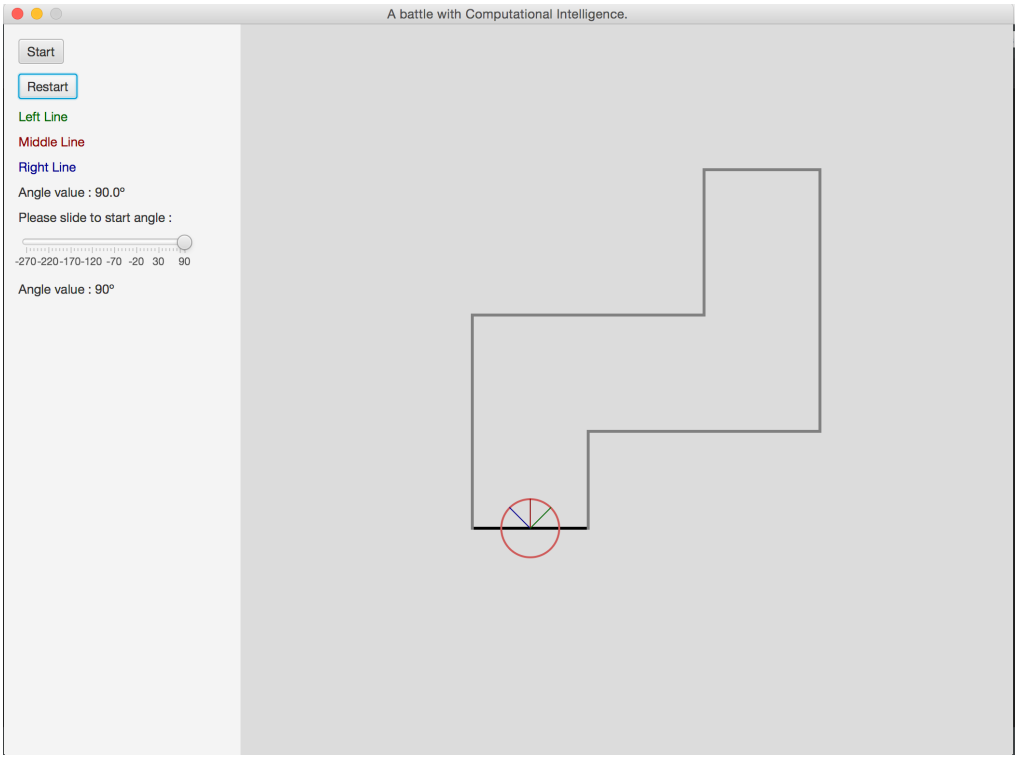
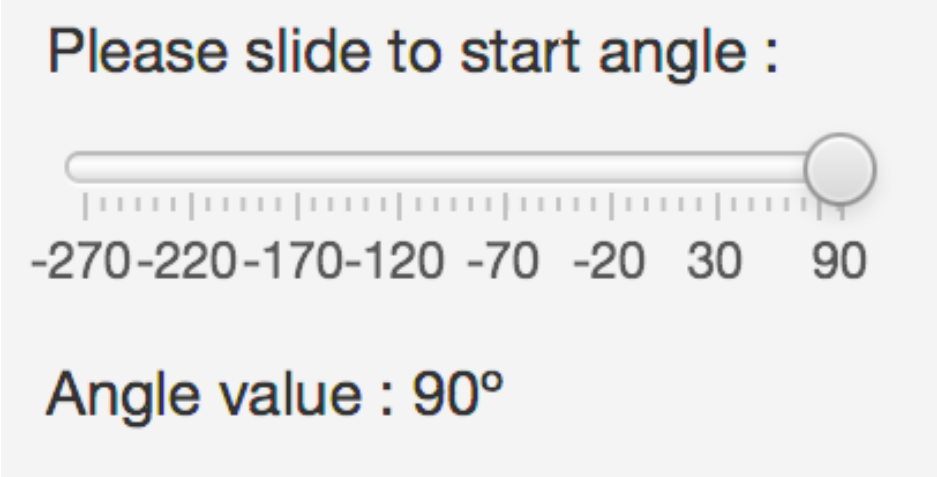
計算型智慧 作業一 說明文件

一、程式執行說明

1. 執行cihw1.jar

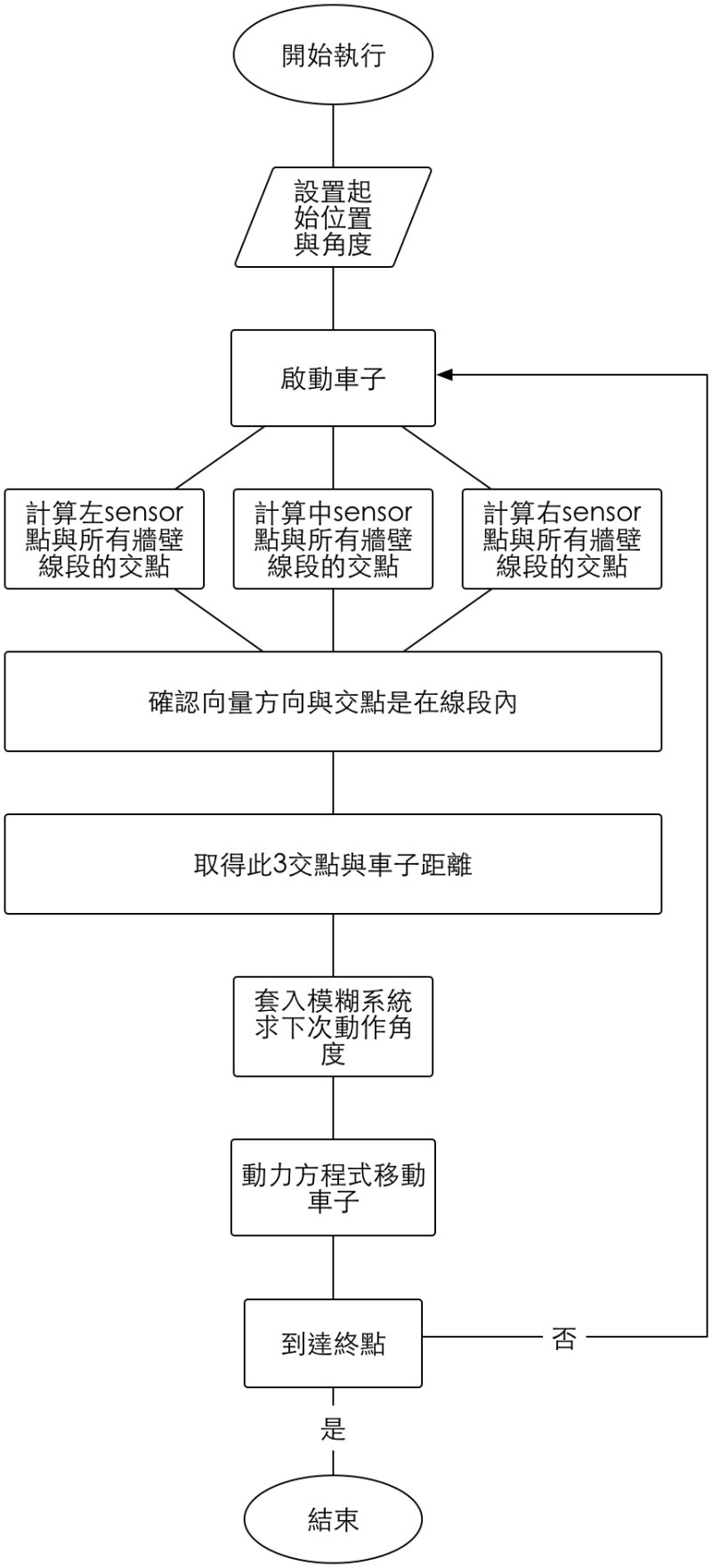


2. 設定起始位置 (直接點選賽道)
3. 拖曳選擇初始角度



4. 點選start按鈕
5. 可以按restart重新開始

二、程式簡介


開始執行
設置起始位置與角度
啟動車子
計算左sensor點與所有牆壁線段的交點
計算中sensor點與所有牆壁線段的交點
計算右sensor點與所有牆壁線段的交點
確認向量方向與交點是在線段內
取得此3交點與車子距離
套入模糊系統求下次動作角度
動力方程式移動車子
到達終點
否
是
結束

1. 這個實驗我是以Java來撰寫，以JavaFX來做圖形化介面，使用者可以選擇自走車的起始位置與起始角度，按下start後車子會開始移動，並依照3個sensor的距離套入模糊系統調整方向，走到終點後可以按下restart重新開始。

2. 計算與牆壁交點這個部分是用了sensor本身的線性方程式，先與八個牆壁線段取交點，但所有的交點並不是我們所想要的，所以最後在用兩個判斷去濾掉這些交點：

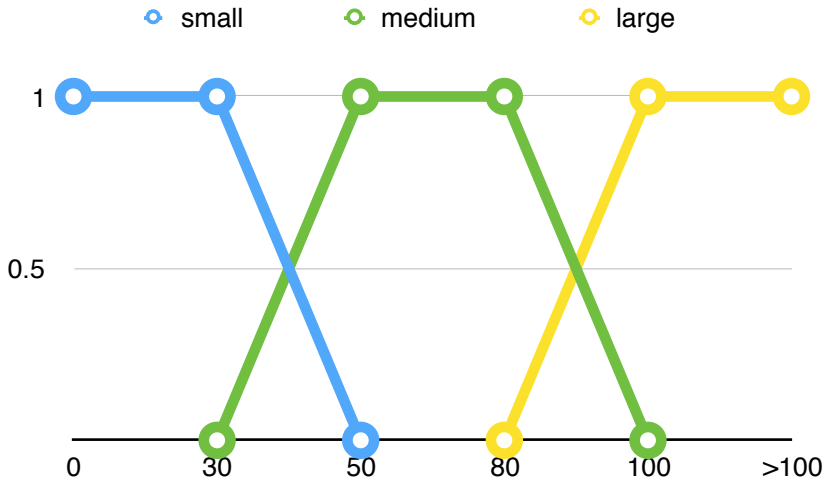
(1) 交點是否有落在牆壁線段上

```
public int checkLineRange(double lineSX, double lineEX, double lineSY, double lineEY, double x, double y) {

    if (lineSX == lineEX) {
        if (y <= lineSY && y >= lineEY) {
            return 1;
        } else {
            return -1;
        }
    } else {
        if (x >= lineSX && x <= lineEX) {
            return 1;
        } else {
            return -1;
        }

    }
}
```

(2) 交點方向是否等同車子面對方向

```
// 計算向量內積，如為正，則代表此交點為sensor面對方向
double ans = itsVectorX * vectorX + itsVectorY * vectorY;
double a = this.x - intersectionX;
double b = this.y - intersectionY;

if (ans > 0) {
    tempDist[i] = Math.sqrt(a * a + b * b);
} else {
    tempDist[i] = Double.MAX_VALUE;
}
```

3. 套入模糊系統找旋轉角度，設27條規則可能，並使用下列函氏圖找距離歸屬函數，最後將規則所設下的角度套入公式計算最後旋轉角度



規則：

```
if (sensorNumber == 3) {
    if (leftS == 0 && middleS == 0 && rightS == 0) {
        angle = checkRank(angle) - 40;
        return angle;

    } else if (leftS == 0 && middleS == 0 && rightS == 1) {
        angle = checkRank(angle) - 40;
        return angle;

    } else if (leftS == 0 && middleS == 0 && rightS == 2) {
        angle = -40;
        angle = checkRank(angle) - 40;
        return angle;

    } else if (leftS == 0 && middleS == 1 && rightS == 0) {
        angle = checkRank(angle) - 40;
        return angle;

    } else if (leftS == 0 && middleS == 1 && rightS == 1) {
        angle = -30;

        angle = checkRank(angle) - 40;
        return angle;

    } else if (leftS == 0 && middleS == 1 && rightS == 2) {
        angle = -40;
        angle = checkRank(angle) - 40;
        return angle;

    } else if (leftS == 0 && middleS == 2 && rightS == 0) {
        angle = 40;
        angle = checkRank(angle) - 40;
        return angle;

    } else if (leftS == 0 && middleS == 2 && rightS == 1) {
        angle = -40;
        angle = checkRank(angle) - 40;
        return angle;

    } else if (leftS == 0 && middleS == 2 && rightS == 2) {
        angle = -40;
        angle = checkRank(angle) - 40;
        return angle;

    } else if (leftS == 1 && middleS == 0 && rightS == 0) {

        angle = checkRank(angle) - 40;
        return angle;

    } else if (leftS == 1 && middleS == 0 && rightS == 1) {
        angle = checkRank(angle) - 40;
        return angle;
```

```
    } else if (leftS == 1 && middleS == 0 && rightS == 2) {
        angle = 10;
        angle = checkRank(angle) - 40;
        return angle;

    } else if (leftS == 1 && middleS == 1 && rightS == 0) {
        angle = 10;
        angle = checkRank(angle) - 40;
        return angle;

    } else if (leftS == 1 && middleS == 1 && rightS == 1) {
        angle = checkRank(angle) - 40;
        return angle;

    } else if (leftS == 1 && middleS == 1 && rightS == 2) {
        angle = -10;
        angle = checkRank(angle) - 40;
        return angle;

    } else if (leftS == 1 && middleS == 2 && rightS == 0) {
        angle = 10;
        angle = checkRank(angle) - 40;
        return angle;

    } else if (leftS == 1 && middleS == 2 && rightS == 1) {
        angle = 30;
        angle = checkRank(angle) - 40;
        return angle;

} else if (leftS == 2 && middleS == 0 && rightS == 0) {
    angle = 30;
    angle = checkRank(angle) - 40;
    return angle;

} else if (leftS == 2 && middleS == 0 && rightS == 1) {
    angle = 10;
    angle = checkRank(angle) - 40;
    return angle;

} else if (leftS == 2 && middleS == 0 && rightS == 2) {
    angle = checkRank(angle) - 40;
    return angle;

} else if (leftS == 2 && middleS == 1 && rightS == 0) {
    angle = 30;
    angle = checkRank(angle) - 40;
    return angle;

} else if (leftS == 2 && middleS == 1 && rightS == 1) {
    angle = 30;
    angle = checkRank(angle) - 40;
    return angle;

} else if (leftS == 2 && middleS == 1 && rightS == 2) {
    angle = checkRank(angle) - 40;
    return angle;
```

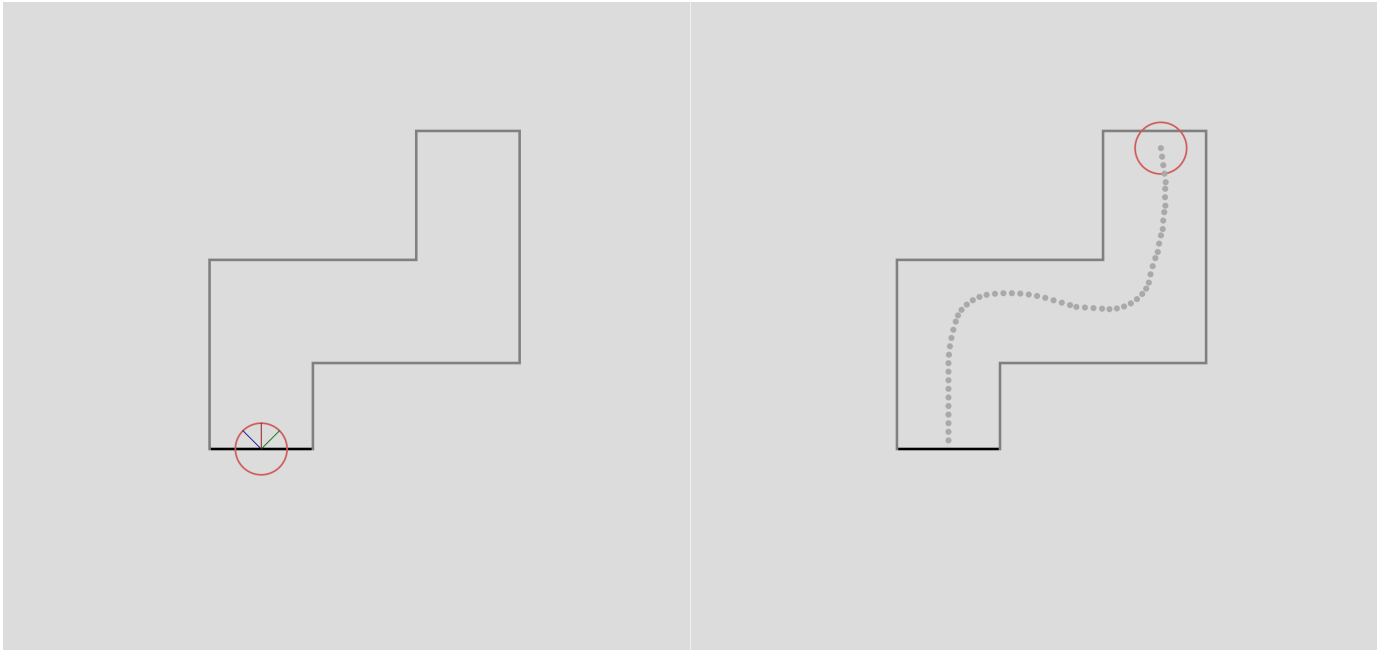
0 - 距離遠

1 - 距離適中

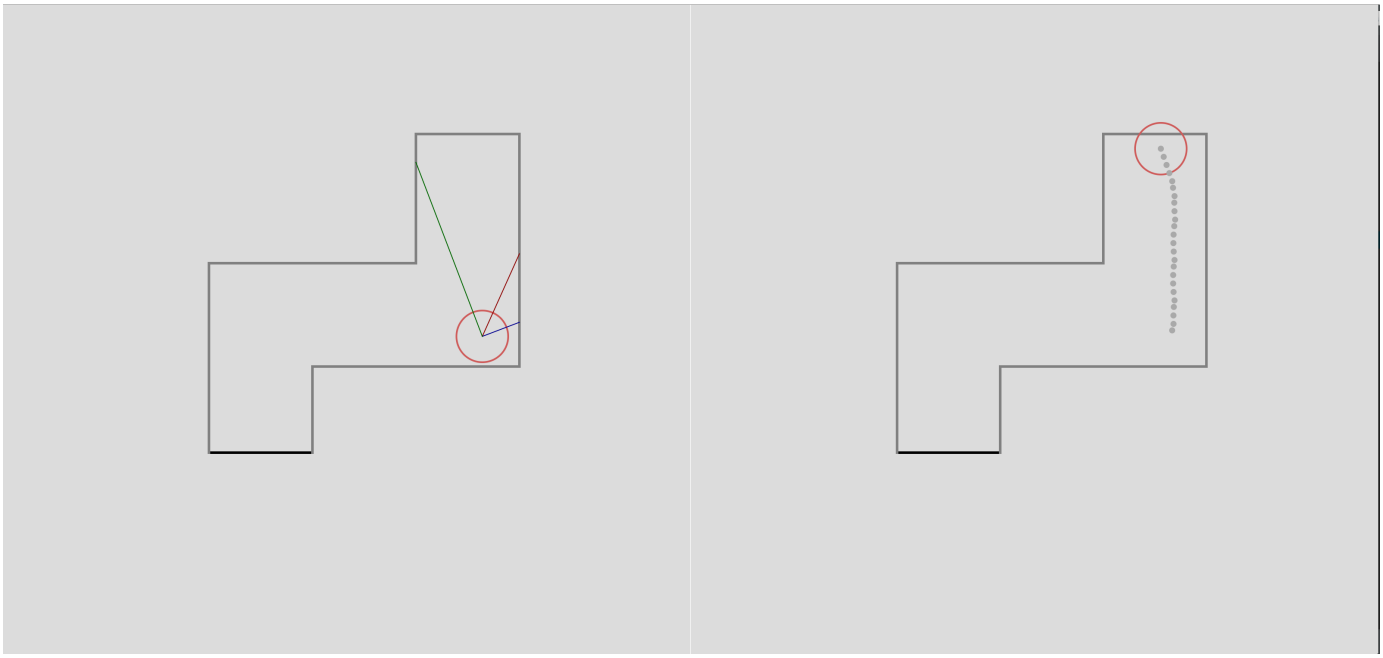
2 - 距離遠

三、實驗結果

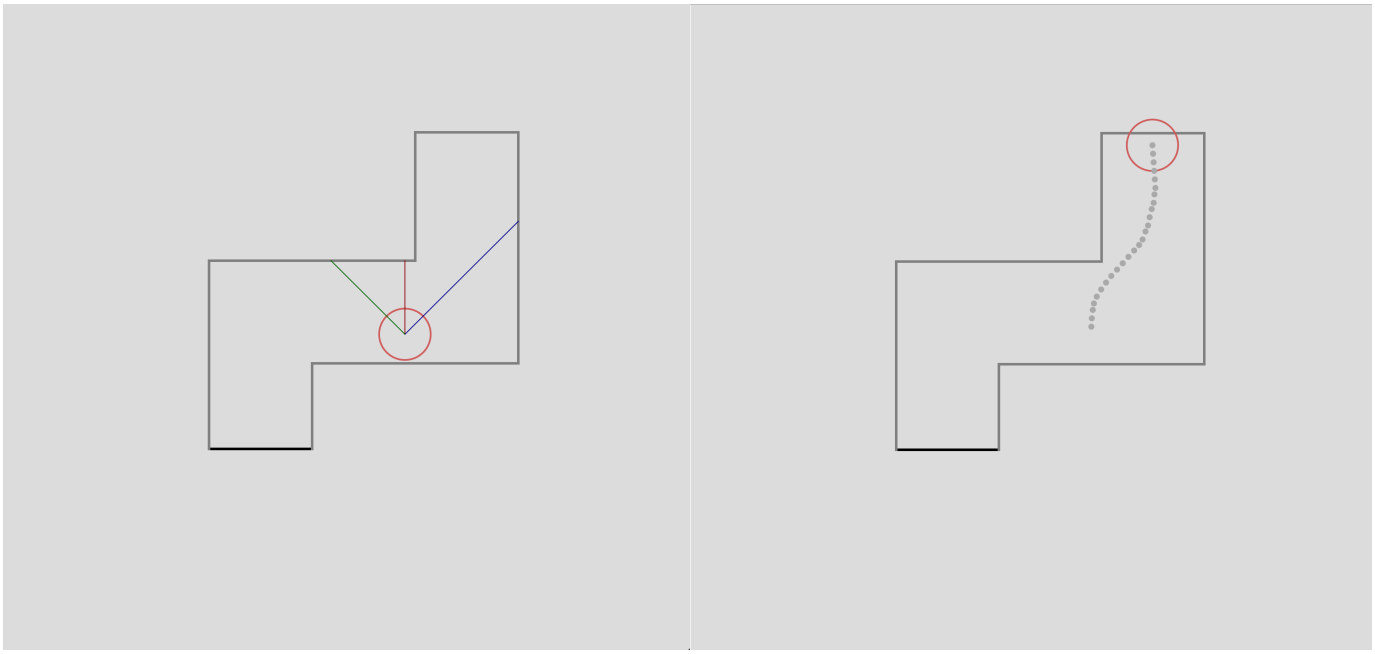
1. 正常起始點



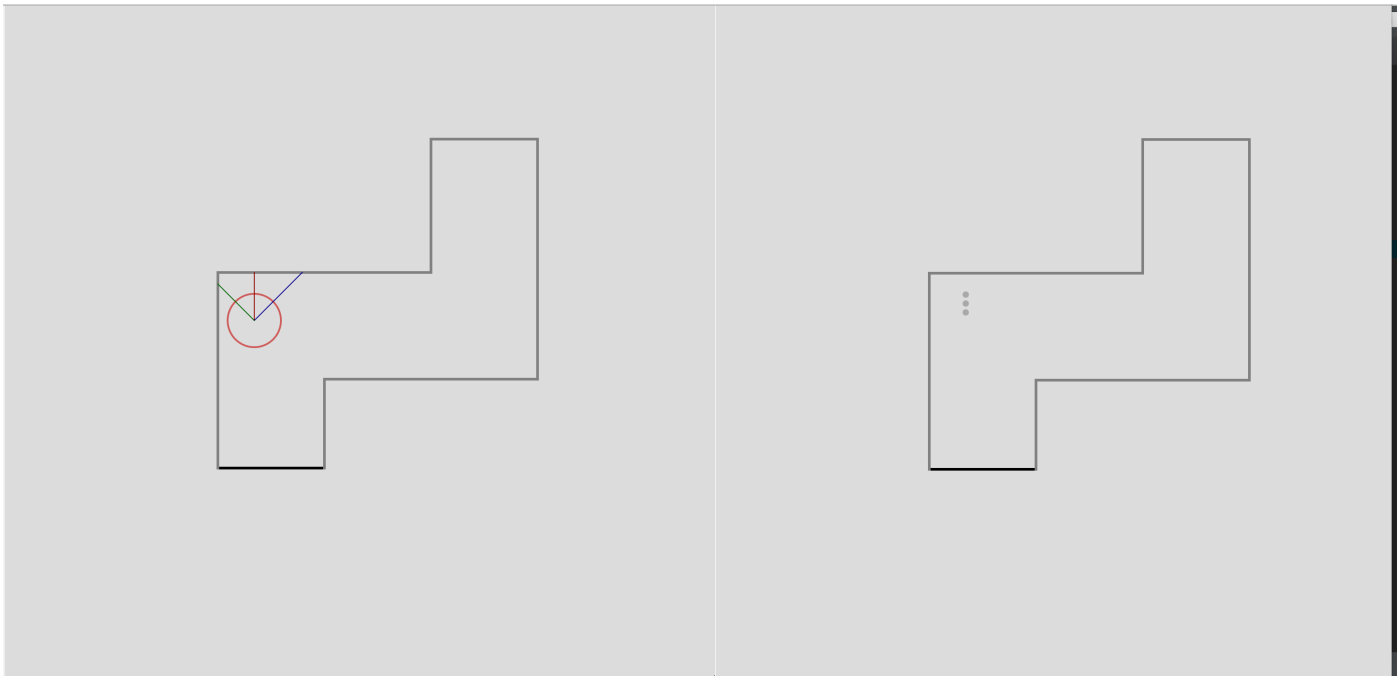
2. 成功



3. 成功



4. 失敗



四、程式結果分析與討論

目前我寫的模糊規則還是有時候會讓自走車撞牆，這部分應該還可以再加強，如果要因應更多賽道這部分是我要補強的，也有些時候車子會直接從直角部分繞過去，這部分的判斷還是要做好，畢竟在現實中不可能發生這樣，還有邊計算邊畫圖型介面的部分還可以做得更好，但礙於不太會開thread多工，要再多參考書本上的用法。