Langton’s ant
Test representation

TripleD.IO - @khofmans

Rules

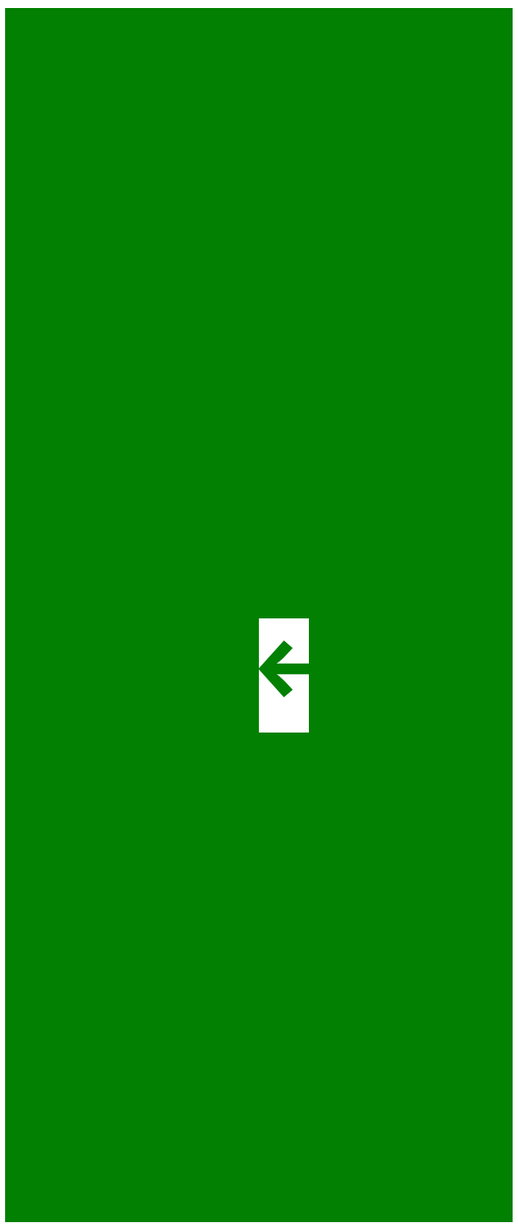
- Black and white squares on a plane
  - One square contains the ant
    - At a **white** square, turn 90° **right**, **flip** the color of the square, move **forward** one unit
    - At a **black** square, turn 90° **left**, **flip** the color of the square, move **forward** one unit

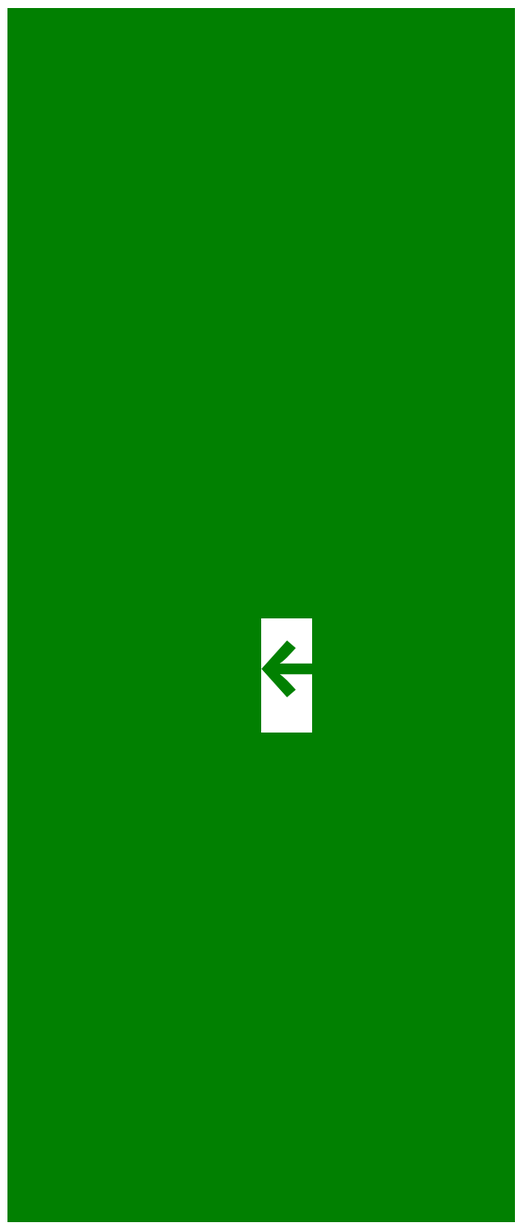
©Wikipedia

Representation

- Black square: 
- White square: ▫
- Ant
- North: ↑
- East: →
- South: ↓
- West: ←

First test
test the presentation





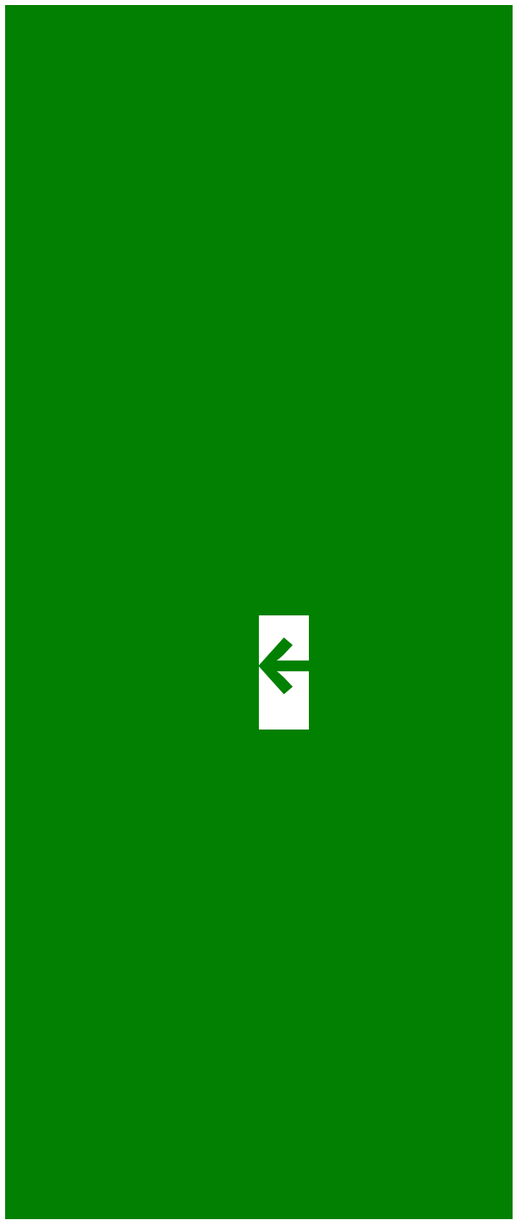
You're not doing anything?

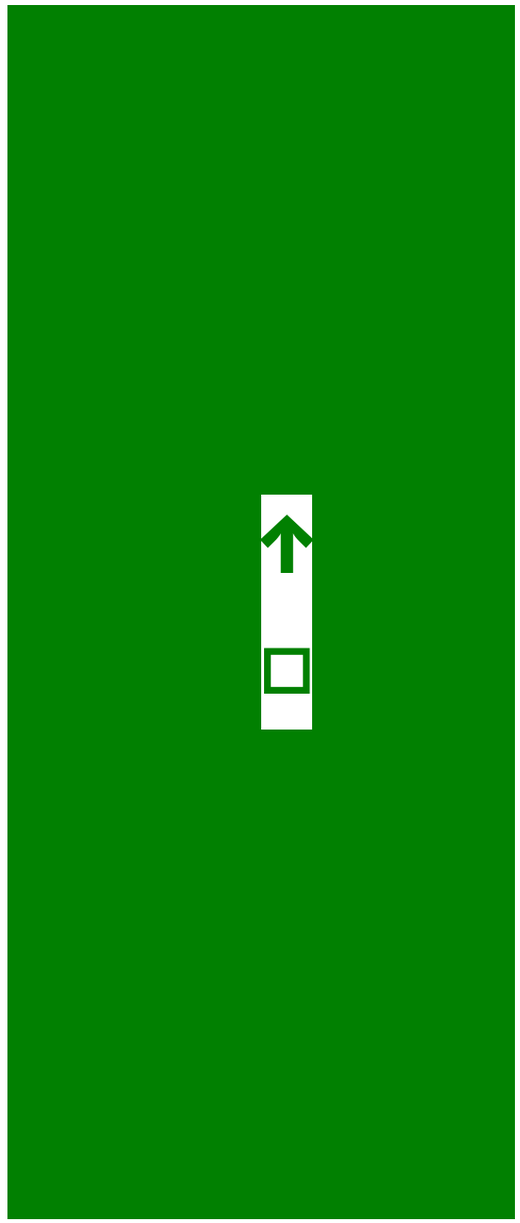
```
fun createGrid(state: String,
               antSquare: SquareContent = BLACK): Grid =
    Grid(ArrayList(state
            .filter { c -> !c.isWhitespace() }
            .map { c -> when (c){
                    '↑' -> Square(ANT, NORTH, antSquare)
                    '←' -> Square(ANT, WEST, antSquare)
                    '→' -> Square(ANT, EAST, antSquare)
                    '↓' -> Square(ANT, SOUTH, antSquare)
                    '■' -> Square(BLACK)
                    '□' -> Square(WHITE)
                   else -> TODO()
                }
        }))


class Grid(var squares: MutableList<Square>) {}
```

And reverse

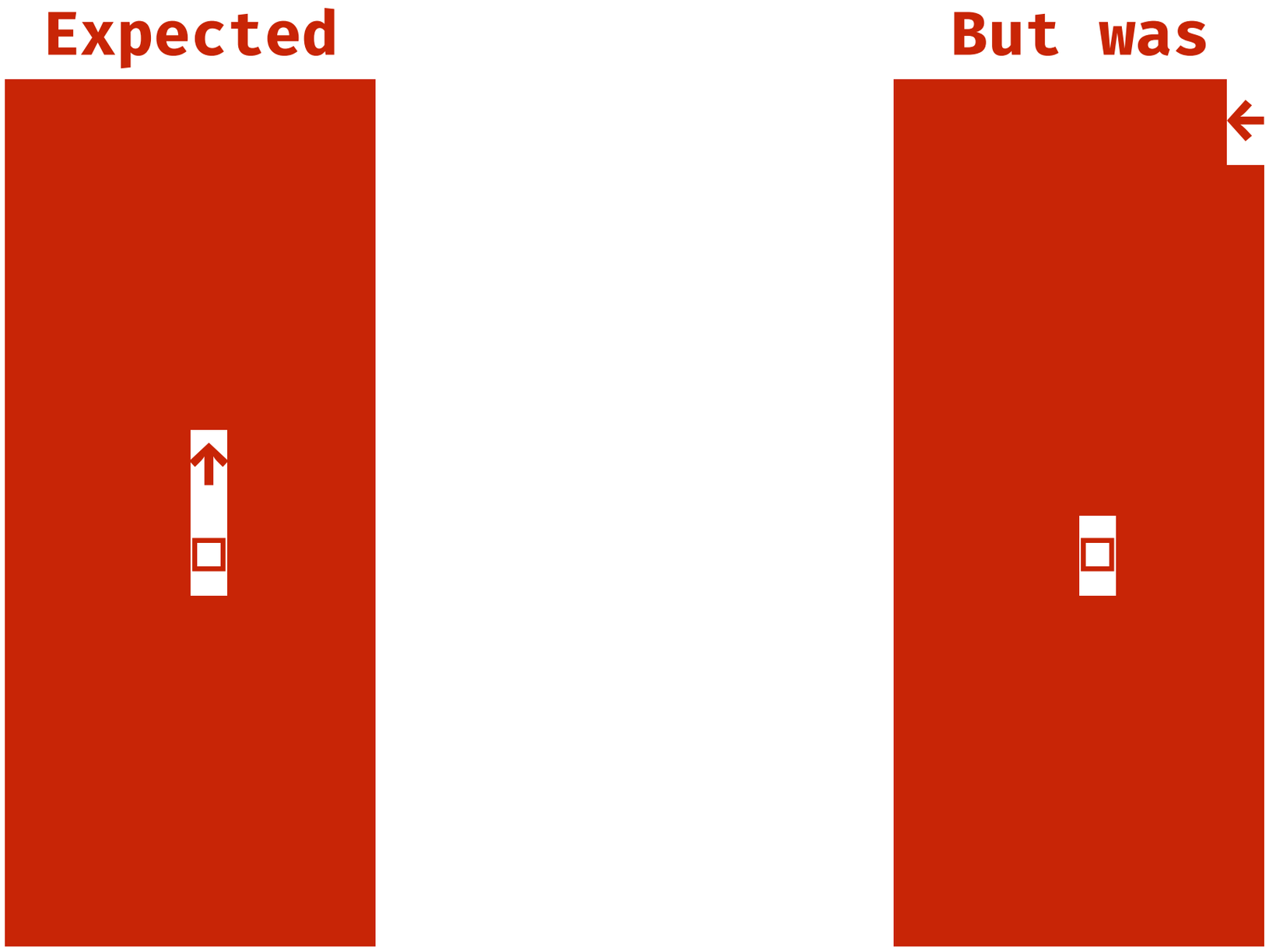
```
fun represent(grid: Grid): String =
grid.squares.map { s -> when (s.content){
    BLACK -> '■'
    WHITE -> '□'
    ANT -> when (s.direction){
        NORTH -> '↑'
        WEST -> '←'
        EAST -> '→'
        SOUTH -> '↓'
        else -> ""
    }
    else -> ""
} }.joinToString("")
```

Second test: start on black





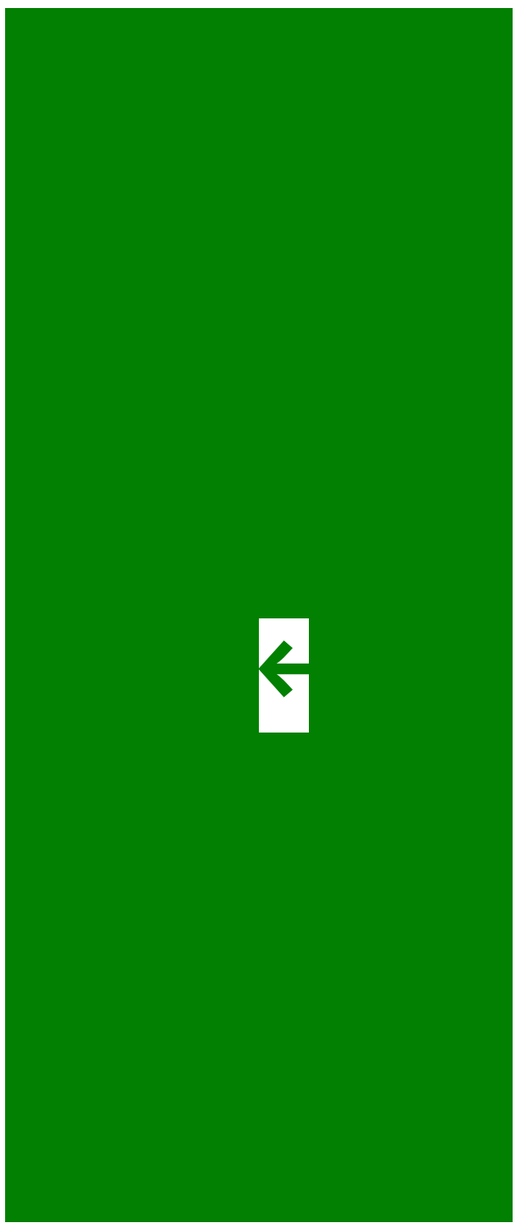
Test feedback

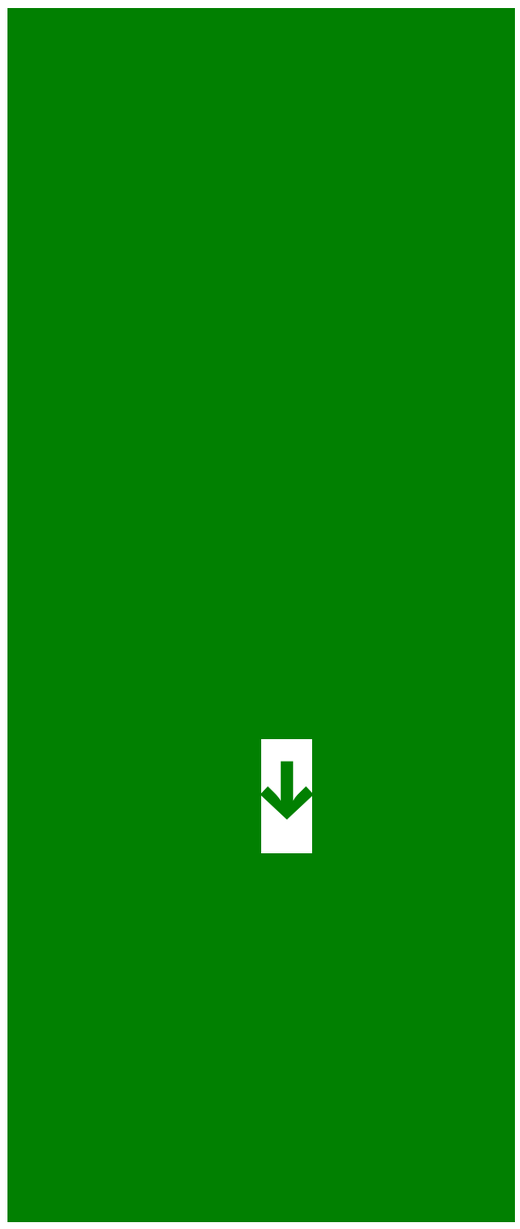


Second test: impl

```
class Grid(var squares: MutableList<Square>) {

    fun tick() {
        val ant = findAnt()
        val indexOfAnt = squares.indexOf(ant)
        squares[indexOfAnt - 10] = Square(ANT, NORTH)
        squares[indexOfAnt] = Square(WHITE)
    }

    private fun findAnt() = squares.find { t->t.content == ANT }
}
```

Third test:
Start on white





Final impl

```
class Grid(var squares: MutableList<Square>) {

    fun tick() {
        val ant = findAnt()
        val location = squares.indexOf(ant)

        flipSquare(ant, location)
        moveToNewLocation(ant, location)
    }

    private fun findAnt(): Square = …

    private fun flipSquare(ant: Square, location: Int) …

    private fun moveToNewLocation(ant: Square, location: Int) …
}
```