**# SC\_project**

**Software Carpentry project**

**Author**

Ride Bu, Yupin Shi, Shengyu Yao

**1. Solution Approach**

Approach with a simple game layout.

Suppose there are…

* Three types of blocks, A (fixed reflect block), B (fixed opaque block, and C (fixed refract block)
* Grid layout is 2\*3

All the permutations of blocks are listed out as the following set:

{('A', 'B', 'C'), ('A', 'C', 'B'), ('B', 'A', 'C'), ('C', 'A', 'B'), ('B', 'C', 'A'), ('C', 'B', 'A')}

Here combinations of grids are list out as the following list:

[((1, 2), (3, 2), (3, 1)), ((1, 2), (3, 2), (2, 1)), ((1, 2), (3, 2), (2, 2)), ((1, 2), (3, 2), (1, 1)), ((1, 2), (3, 1), (2, 1)), ((1, 2), (3, 1), (2, 2)), ((1, 2), (3, 1), (1, 1)), ((1, 2), (2, 1), (2, 2)), ((1, 2), (2, 1), (1, 1)), ((1, 2), (2, 2), (1, 1)), ((3, 2), (3, 1), (2, 1)), ((3, 2), (3, 1), (2, 2)), ((3, 2), (3, 1), (1, 1)), ((3, 2), (2, 1), (2, 2)), ((3, 2), (2, 1), (1, 1)), ((3, 2), (2, 2), (1, 1)), ((3, 1), (2, 1), (2, 2)), ((3, 1), (2, 1), (1, 1)), ((3, 1), (2, 2), (1, 1)), ((2, 1), (2, 2), (1, 1))]

By combining the set and the list above using Cartesian Product, all the possible combinations of blocks and grids are obtained and stored. The 120 possible solutions can be viewed as:

(('A', 'B', 'C'), ((1, 2), (3, 2), (3, 1)))

(('A', 'B', 'C'), ((1, 2), (3, 2), (2, 1)))

...

(('C', 'B', 'A'), ((1, 2), (3, 2), (3, 1)))

(('C', 'B', 'A'), ((1, 2), (3, 2), (2, 1)))

(('C', 'B', 'A'), ((1, 2), (3, 2), (2, 2)))

The check\_position() function is used to verify if the blocks A, B, and C are able to reflect, refract, or block the lights, so that the “lazor” will go through all the target points on the layout at the same time.

In the function check\_position(), the points which the “lazor” passes are tracked. These include all the points and the directions of the “lazor” when it goes through the point. The program breaks when the “lazor” is either blocked or going out of the layout. All the points are compared with the locations of pre-defined targets. If all the targets are matched, then the program returns the locations of each block, respectively.

**2. How to play**

1. For the given game layout

Solution can be obtained directly by running the following .py file in commander:

Python final\_solution.py

1. For other game layouts

Please update the content between the ‘’ in the following lines with the name of the corresponding .bff file of your choice. Please update it before running the file.

if \_\_name\_\_ == "\_\_main\_\_":

find\_solution('***mad\_6.bff***')