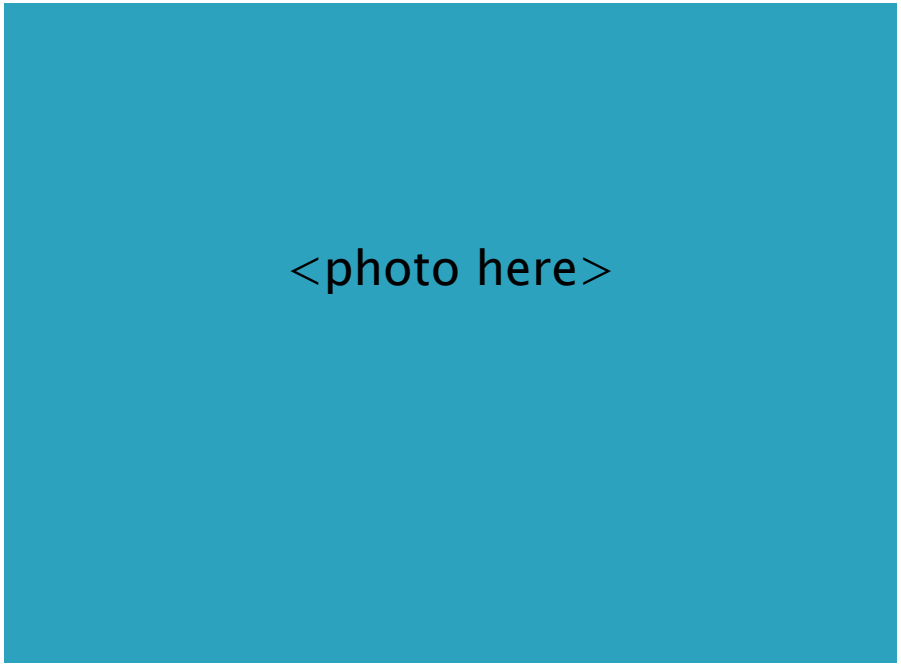
當 Python 遇上魔術方塊

戴嘉駿 darkgerm <darkgerm@gmail.com>

關於 darkgerm

- 目前就讀於交通大學資工系，是個熱愛 Python 及各種資訊技術的大學生
- 平時的休閒活動喜歡玩各種益智遊戲，像是魔術方塊、紙上謎題、和各類益智玩具。
- 於 2011 年底開始參加 Python 新竹地區的使用者社群(PyHUG)，現在是 PyHUG 的協辦單位幹部，也曾在 PyHUG 上給過一個與 ctypes 相關的 talk。



Outline

- 1. 魔術方塊的基本認識
  - 1.1 魔方的介紹
  - 1.2 魔方的解法
- 2. pyRubiks 程式架構
  - 2.1 什麼是 pyRubiks – 動機及目的
  - 2.2 讀取 – 讓電腦「看到」魔方
  - 2.3 儲存 – 魔方的資料結構
  - 2.4 尋解 – 找出其解法
  - 2.5 顯示 – 畫出一顆魔方
  - 2.6 demo !
  - 2.7 pyRubiks 其他延伸應用
- 3. 一些魔方記錄及軼事

1. 魔術方塊的基本認識

- 1.1 魔方的介紹
- 1.2 魔方的解法

1.1 魔方的介紹

- 1974 年匈牙利建築學教授 Rubik Ernő 發明
- 1980 年代大流行
- 變化數：
  - 角排列 * 角旋轉 * 邊排列 * 邊旋轉 / 反向排列

$$\frac{8! \times 3^7 \times 12! \times 2^{11}}{2} = 43{,}252{,}003{,}274{,}489{,}856{,}000$$

  - 37 EB !
    - KB MB GB TB PB(PetaByte) EB(ExaByte)

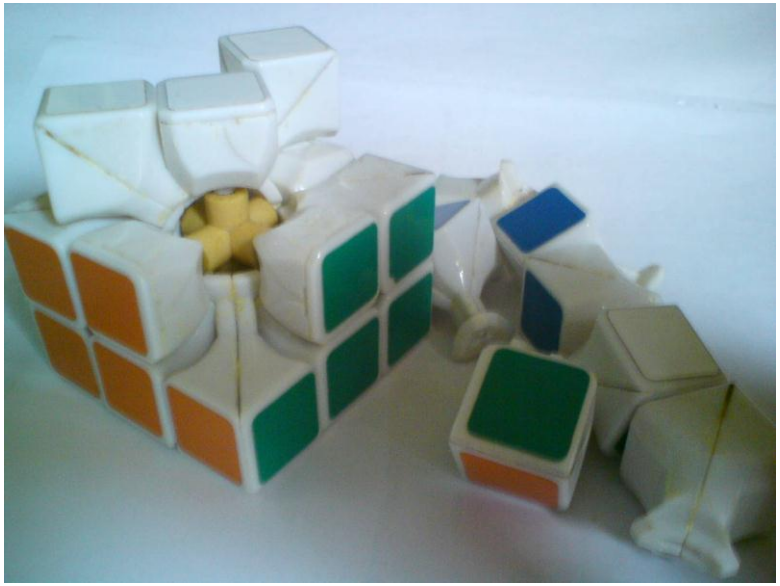
1.2 魔方的解法

- 爆力解 (?)
- 我的解法
- 阿公的解法
- layer-by-layer
- 速解魔方(Fridrich Method, CFOP system)
- 八角定位(corner first)
- 8355 解法 (出自台灣！)
- 盲解魔方

1.2 魔方的解法

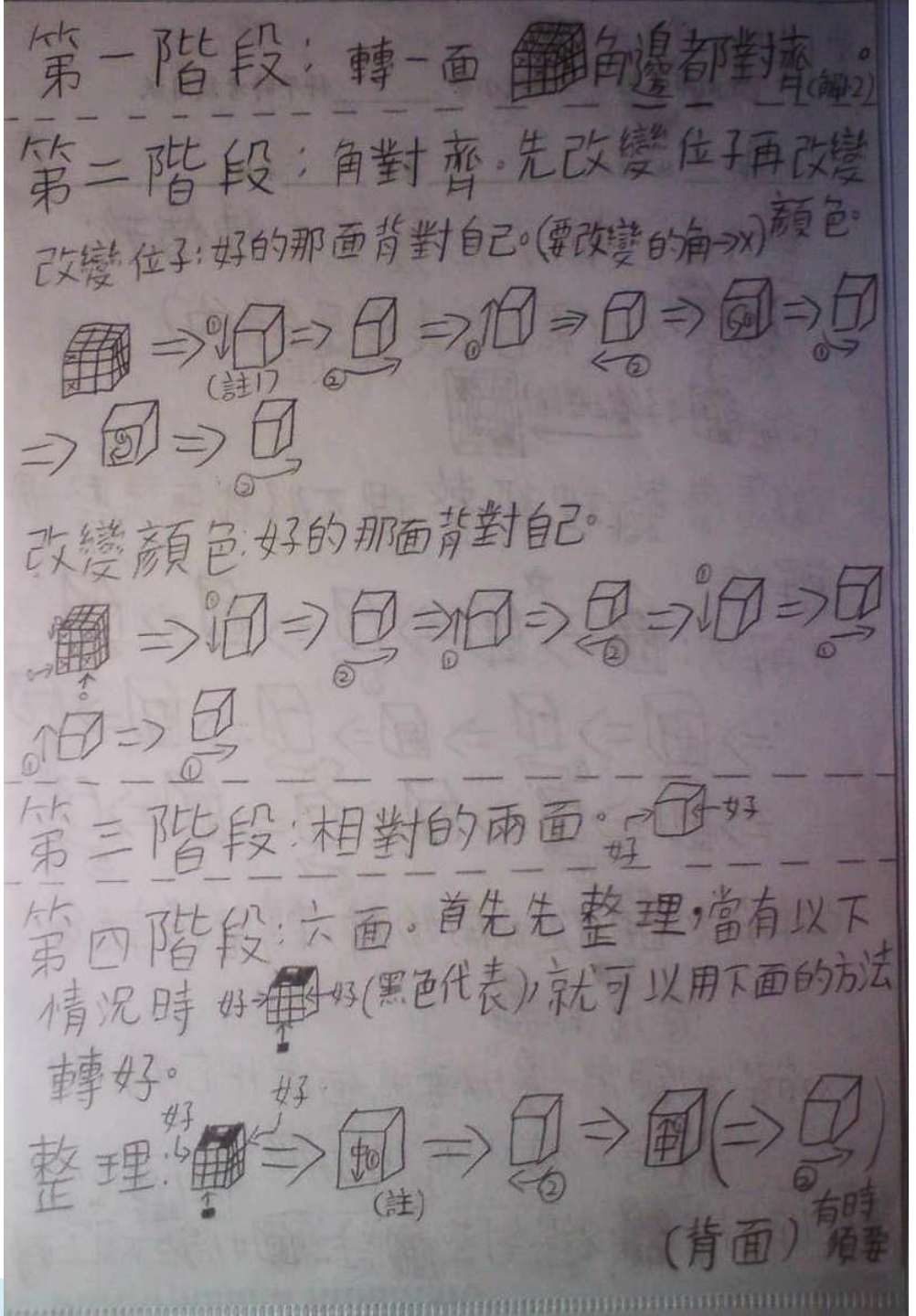
- 爆力解 (?)
  - 重貼貼紙
  - 拆開重組

FAIL !



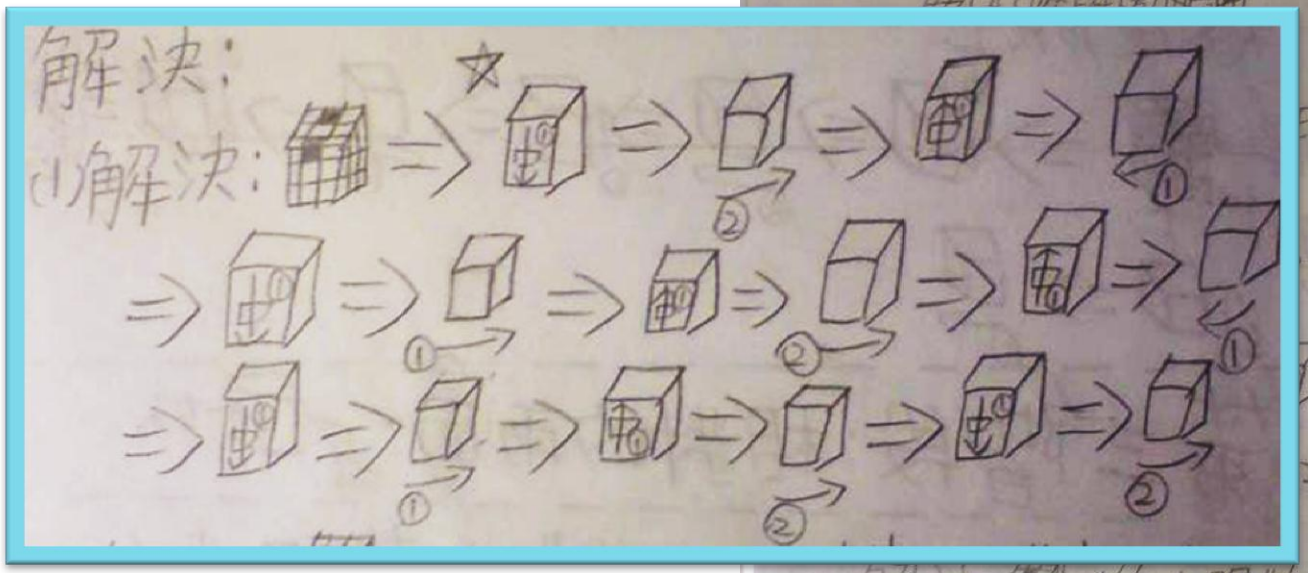
1.2 魔方的解法

- 我的解法



1.2 魔方的解法

- 我的解法



台南市 區 國民小學 科平時考試用紙

年 班 姓名 座號 成績

整理完後,可能發生三種情形:

(1)好好(黑色代表還是錯的)

理不好或無法整理。

(2)解決:從★開始轉,轉完後就會變成(1)。

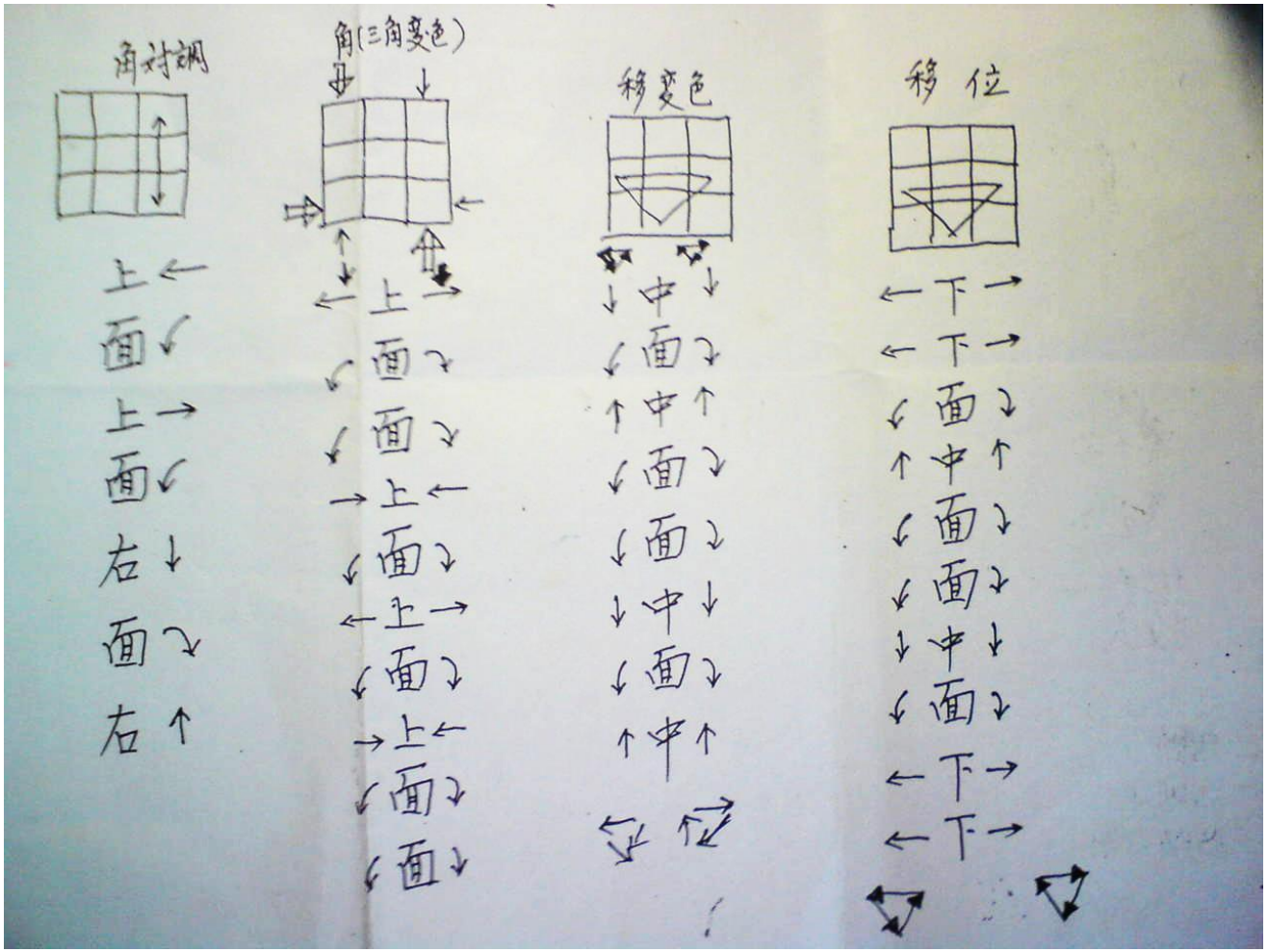
(3)解決:鎖定一邊(必要時一面)並把它轉好,就會形成(1)或(2)情形。

註1:面次數方向

解:1.對齊 2.邊對齊 註2:面中間往下或上轉

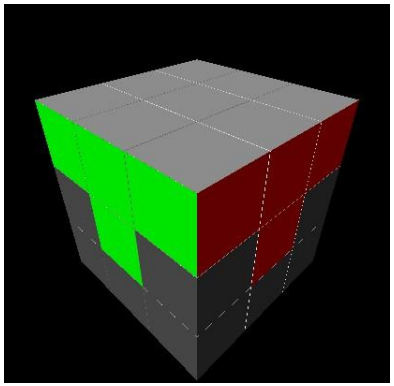
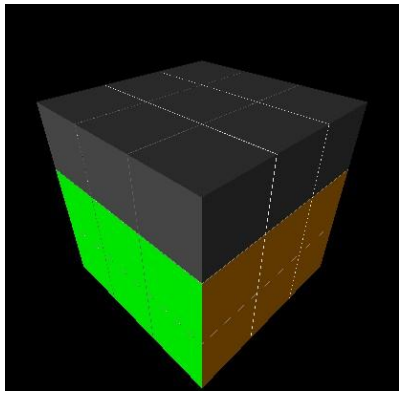
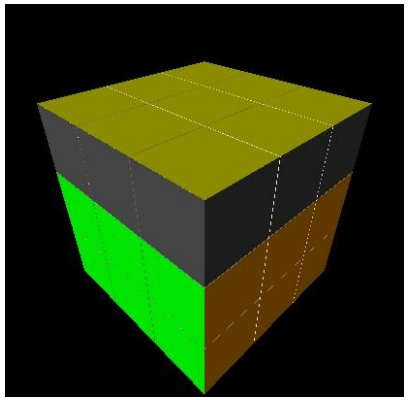
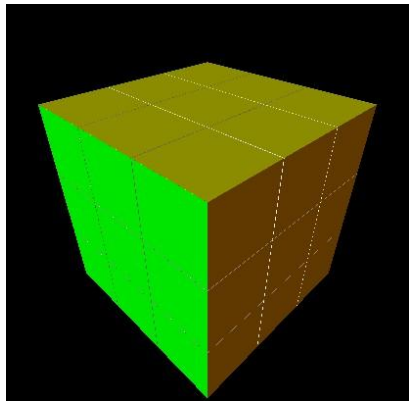
1.2 魔方的解法

- 阿公的解法



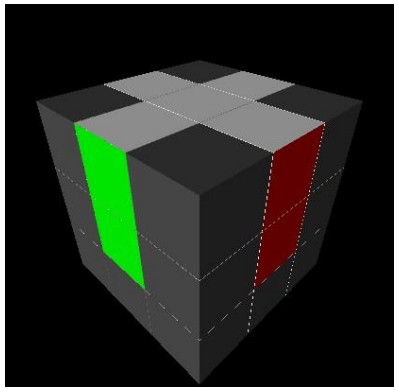
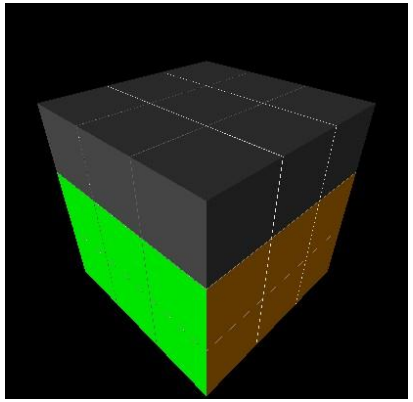
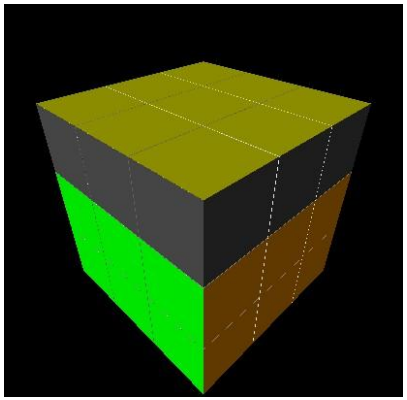
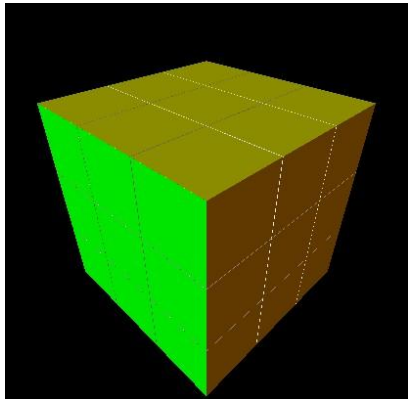
1.2 魔方的解法

- layer-by-layer
  - 一層一層解

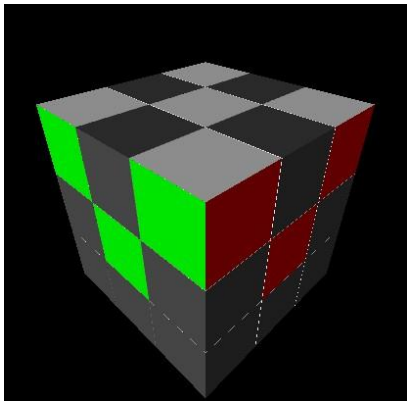
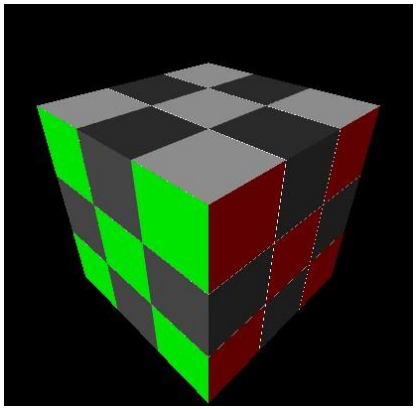
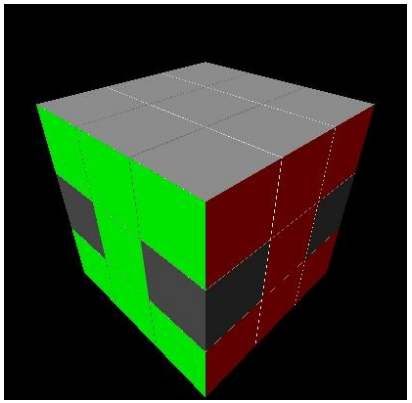
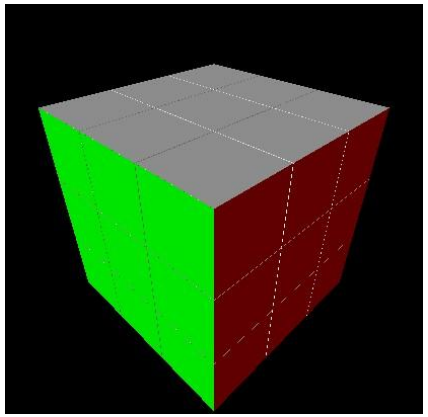
1.2 魔方的解法

- 速解魔方(Fridrich Method, CFOP system)
  - layer-by-layer 的進階版
  - 人轉起來速度最快，目前在比賽最多人使用
  - 公式眾多：第三階段有 57 個、第四階段有 21 個

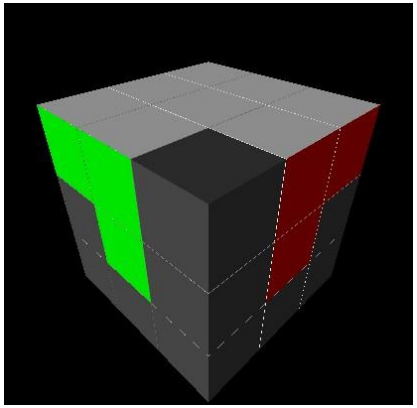
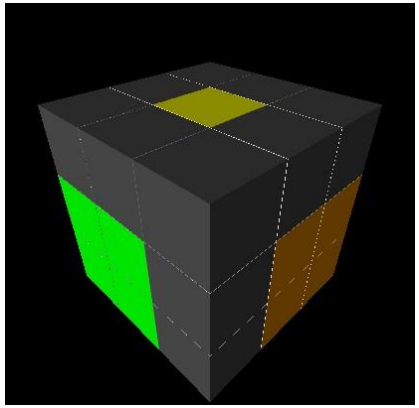
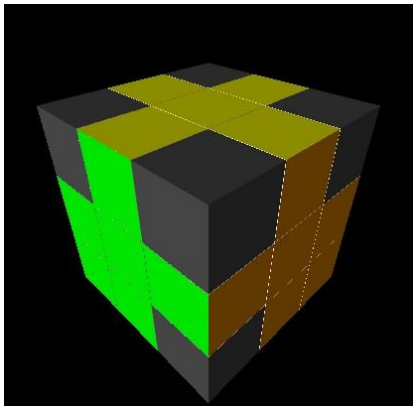
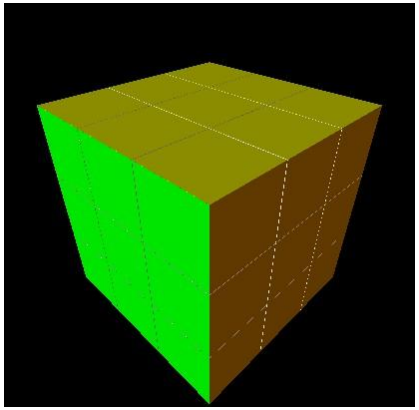
1.2 魔方的解法

- 八角定位(corner first)
  - 「角先」轉法
  - 與我發現的解法相同！

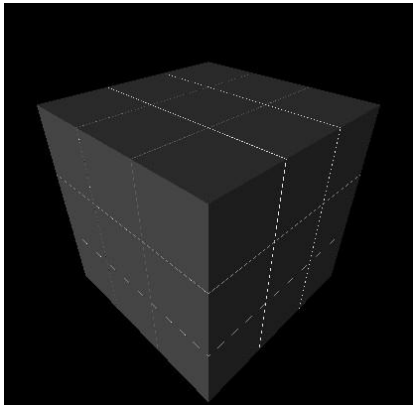
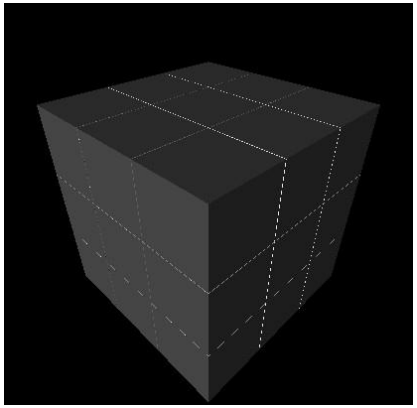
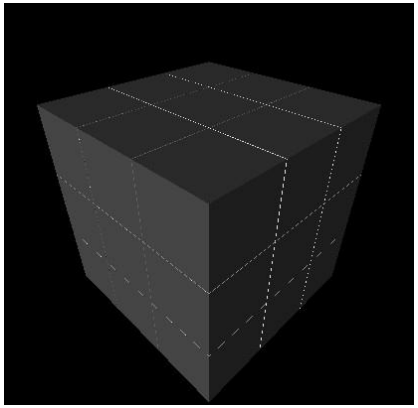
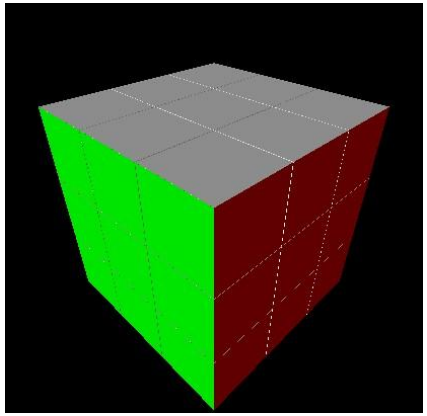
1.2 魔方的解法

- 8355 解法 (出自台灣！)
  - 許技江老師所規劃出來的解法
  - 強調沒有公式！靠理解的方式解魔方

1.2 魔方的解法

- 盲解魔方
  - 把整個方塊背下來，頭腦的 memory 要夠

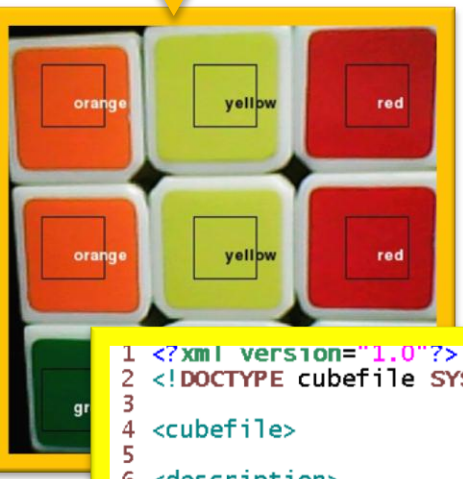
2. pyRubiks 程式架構

- 2.1 什麼是 pyRubiks – 動機及目的
- 2.2 讀取 – 讓電腦「看到」魔方
  - SimpleCV
- 2.3 儲存 – 魔方的資料結構
  - numpy
- 2.4 尋解 – 找出其解法
- 2.5 顯示 – 畫出一顆魔方
  - VPython
- 2.6 demo !
- 2.7 pyRubiks 其他延伸應用

2.1 什麼是 pyRubiks – 動機及目的

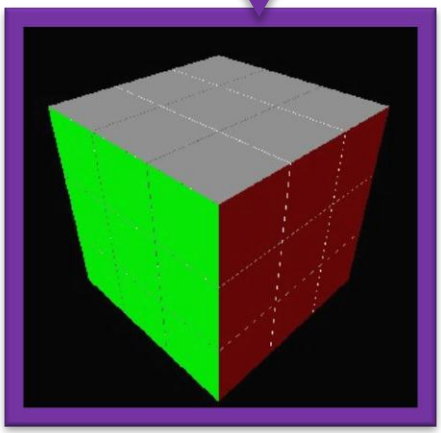
- 一個用 Python 寫的魔術方塊程式
  - 要能有像動畫般的轉方塊畫面
  - 要能夠自己找到解
  - 最好還要可以讀取一顆方塊
- 動機
  - 出自對魔方的興趣
  - 看到網路上有人用樂高做「轉魔方機器」，想自己也做一個
- https://bitbucket.org/darkgerm/pyrubiks/src

real Rubik's cube → SimpleCV → xml file → XML parser → class cube → cubeSolver → list of moves → VPython → display it




```xml
<?xml version="1.0"?>
<!DOCTYPE cubefile SYSTEM "c
<cubefile>
<description>
    This is a test file.
</description>
<action name="test 1">
    <description> raw cube.
    <init>
        GYG
        BWY
        GWY
        YGB ROR WGY RYO
        BGY RRO BBO WOR
        YRW BGO BOO WWO
        BWR
        BYR
        GGW
    </init>
</action>
</cubefile>
```

```python
class Cube(object):

    def __init__(self):
        '''A Cube is too complicated. Do not use __in
        the __init__ only generate a complete cube.
        '''
        self.pieces = np.array([
            [
                [Piece( 0), Piece( 9), Piece(18)],
                [Piece( 3), Piece(12), Piece(21)],
                [Piece( 6), Piece(15), Piece(24)],
            ],

            [
                [Piece( 1), Piece(10), Piece(19)],
                [Piece( 4), Piece(13), Piece(22)],
                [Piece( 7), Piece(16), Piece(25)],
            ],

            [
                [Piece( 2), Piece(11), Piece(20)],
                [Piece( 5), Piece(14), Piece(23)],
                [Piece( 8), Piece(17), Piece(26)],
            ],
        ])
```

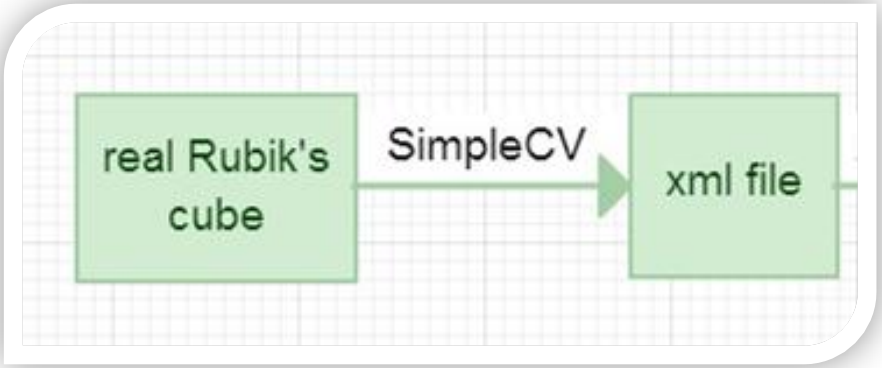


```
['U2', 'F2', 'D2', 'L', 'D2', 'R', "B'", "R'",
"F'", 'L', 'U', "L'", 'U', "F'", 'U', 'F', 'U2',
'B', 'U', "B'", "U'", 'L', "U'", "L'", 'U', 'L'
F'", 'L2', "F'", "L'", 'F2', 'R', 'L', 'F', "U'"
, "F'", "U'", "F'", 'L', 'F', "L'"]
```

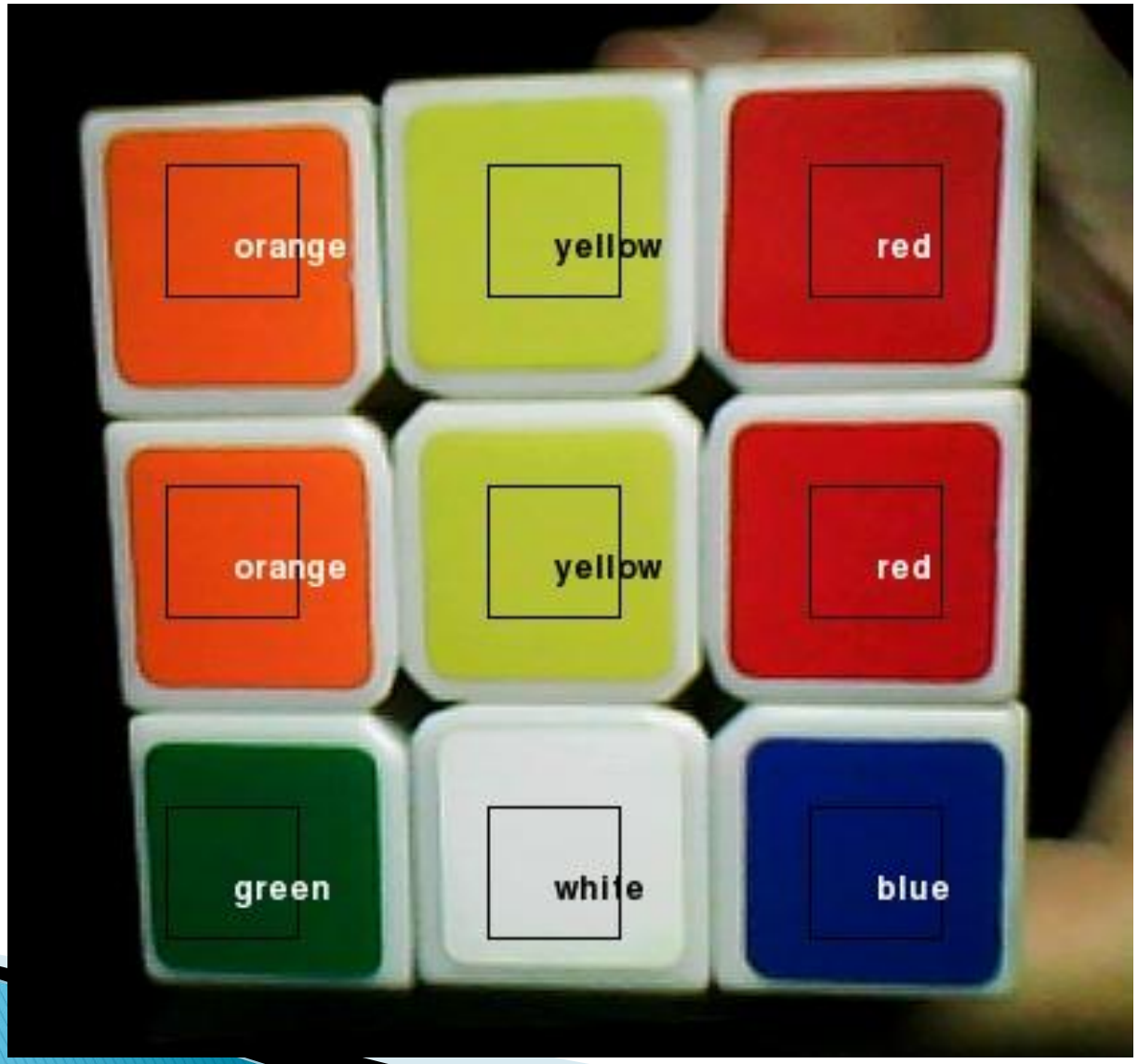
2.2 讀取 – 讓電腦「看到」魔方

- 使用 SimpleCV 做攝影機的控制及顏色的判別
- 週遭燈光的影響，有些顏色常常會誤判...
  - 黃色 vs 白色
  - 紅色 vs 橙色
- SimpleCV 在 mac 上非常難安裝orz


```
cam = Camera()
img = cam.getImage()
img.show()
```



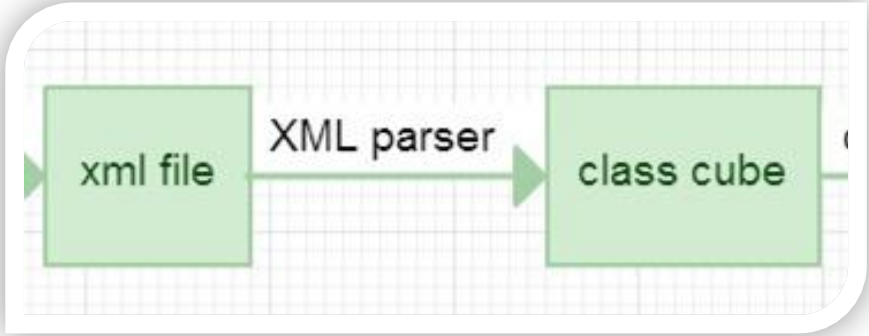
2.2 讀取 – 讓電腦「看到」魔方



2.3 儲存 – 魔方的資料結構

- 一顆 3D 的魔方，如何用文字描述？
  - 分別寫出每個小方塊的顏色
    - 魔方是 3D 的，太難閱讀
  - 以展開圖的形式畫出魔方
    - 雖然無法表示出一個小方塊的狀態，但可讀性高很多

```
   GYG
   BWY
   GWY
YGB ROR WGY RYO
BGY RRO BBO WOR
YRW BGO BOO WWO
   BWR
   BYR
   GGW
```



2.3 儲存 – 魔方的資料結構

- 一個表示魔方的字串，要用什麼檔案格式儲存？
  - JSON
    - 字串不能換行，可讀性消失了
  - ini
    - 字串可換行
    - 魔方是資料，不是設定檔，感覺很怪
      - 寫程式也是需要 feeling 的 ☺
  - XML
    - 字串可換行
    - DTD(Document Type Definition)

```
<!ELEMENT cubefile (description?, action+)>
<!ELEMENT action (description?|moves|init|goal?)>
<!ATTLIST action
  name   CDATA   #IMPLIED
>
```

2.3 儲存 – 魔方的資料結構

- 要用什麼資料結構儲存魔方？
  - numpy.array
    - 可以輕鬆的取出某個特定的面，同時也可以直接賦新值

```
def _F_getter(self):
    return np.rot90(self.pieces[:, :, 2], 1)
def _F_setter(self, x):
    self.pieces[:, :, 2] = np.rot90(x, 3)
F = property(_F_getter, _F_setter)

def _B_getter(self):
    return np.rot90(self.pieces[:, ::-1, 0], 3)
def _B_setter(self, x):
    self.pieces[:, ::-1, 0] = np.rot90(x, 1)
B = property(_B_getter, _B_setter)

def _U_getter(self):
    return np.rot90(self.pieces[::-1, 2, :], 3)
def _U_setter(self, x):
    self.pieces[::-1, 2, :] = np.rot90(x, 1)
U = property(_U_getter, _U_setter)
```

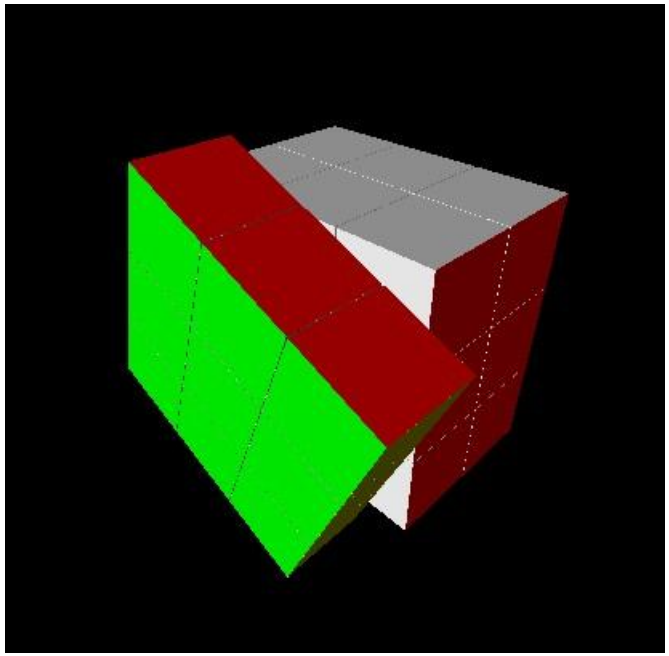
2.3 儲存 – 魔方的資料結構

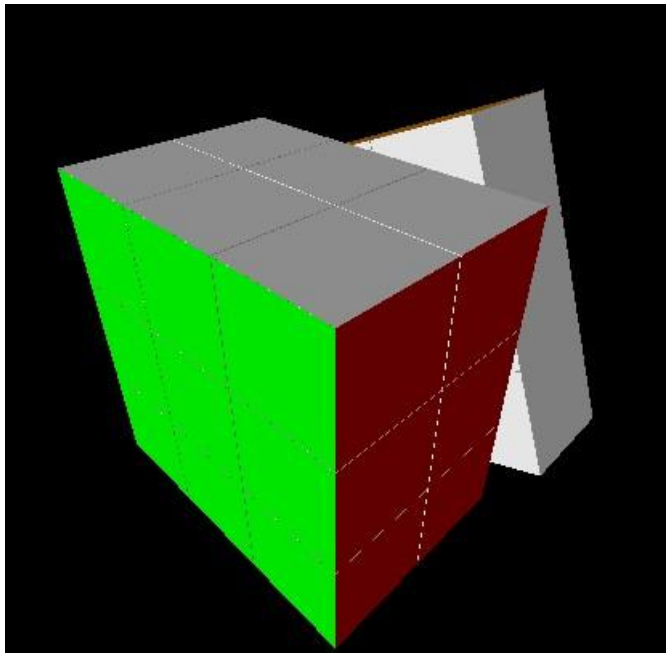
- 魔方能有什麼動作(method)？
  - turn()
    - Singmaster notation
    - L U L‘ y R2 u‘ R U‘ R‘ U R‘ u R2

```
def turn(self, moves):
    '''Singmaster notation: FBUDLRfbudrlxyz + "2'"
    "MES" extension also available.
    ignore all other character.

    "MES" extension:
    M (Middle): the layer between L and R, turn direction as L (top-down)
    E (Equator): the layer between U and D, turn direction as D (left-right)
    S (Standing): the layer between F and B, turn direction as F
    '''
```

2.3 儲存 – 魔方的資料結構

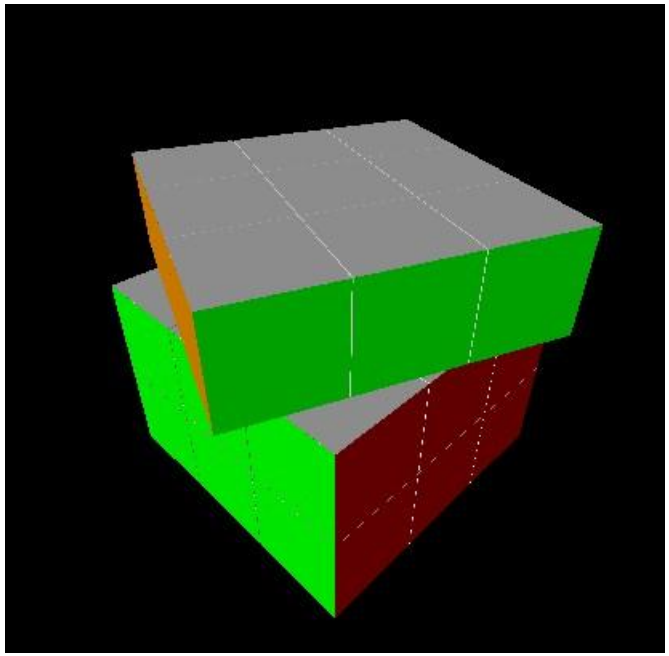
- Singmaster notation (綠色面向自己)
  - F (Front)　　B (Back)

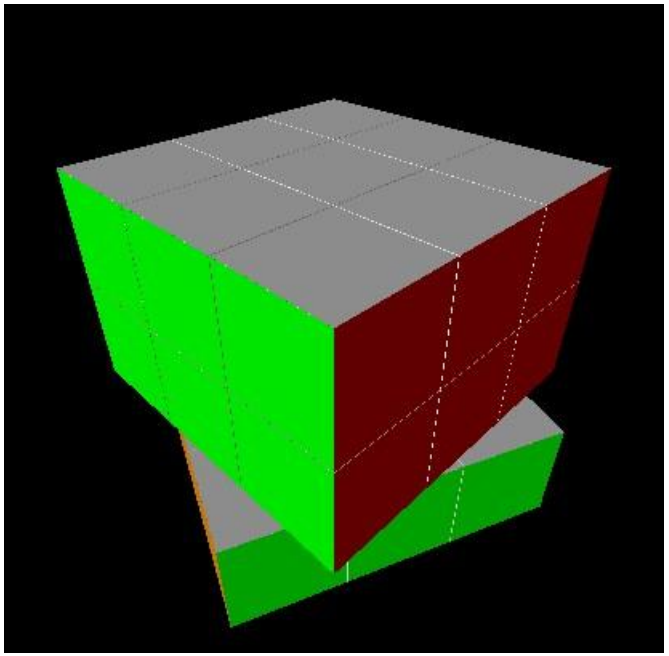




2.3 儲存 – 魔方的資料結構

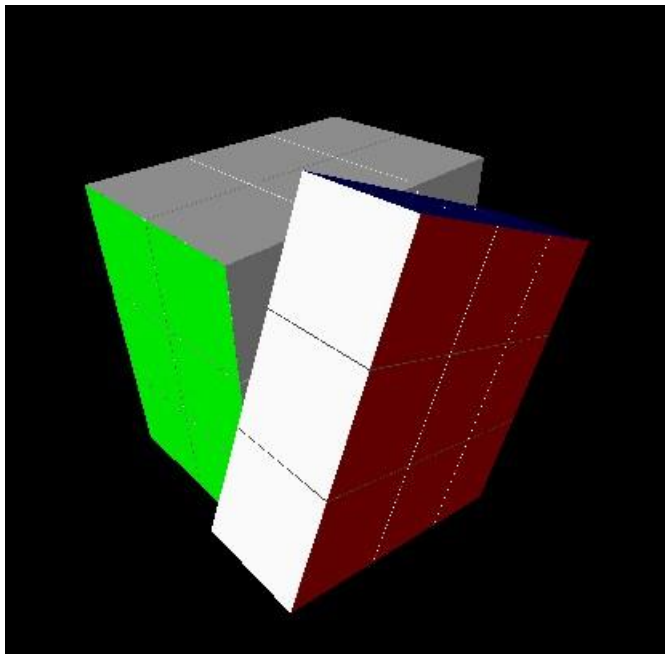
- Singmaster notation (綠色面向自己)
  - U (Up)
  - D (Down)

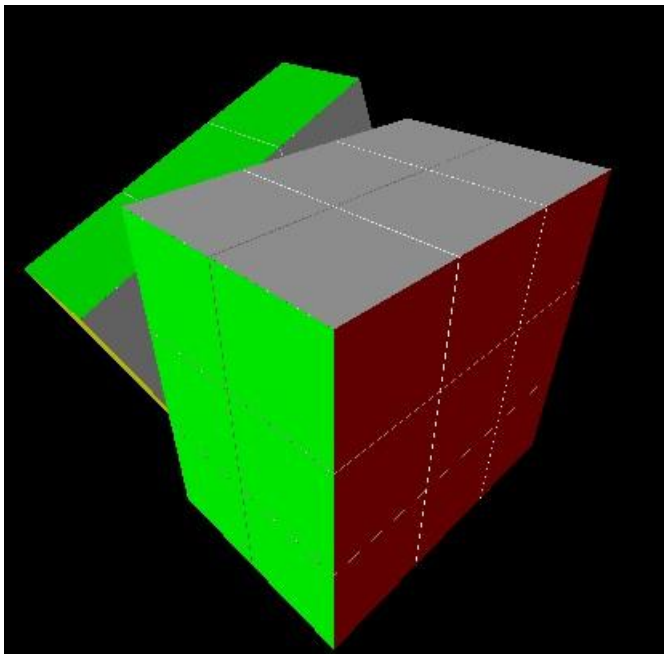




2.3 儲存 – 魔方的資料結構

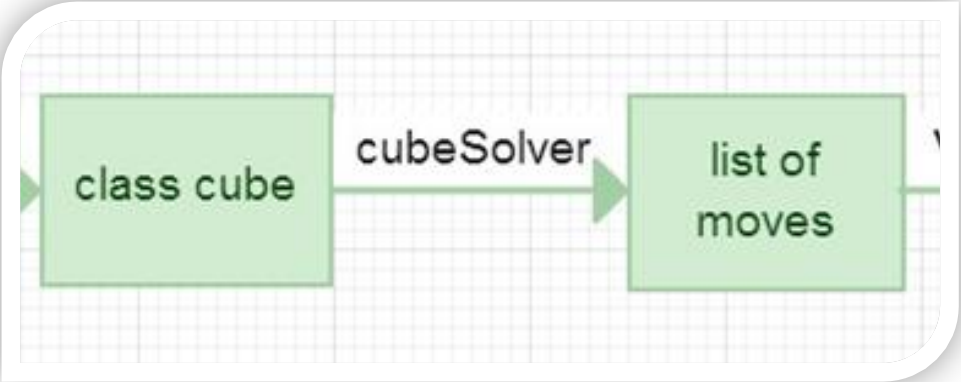
- Singmaster notation (綠色面向自己)
  - R (Right)　　L (Left)





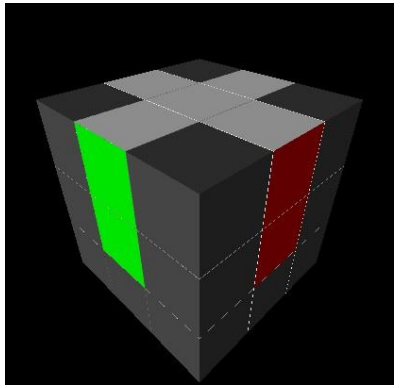
2.4 尋解 – 找出其解法

- 使用 Fridrich Method
  - 最多人使用的解法，但尚未看過有程式做過
- Fridrich Method 分成四個階段 (CFOP)
  - cross
  - F2L (First 2 Layer)
  - OLL (Orientation of Last Layer)
  - PLL (Permutation of Last Layer)



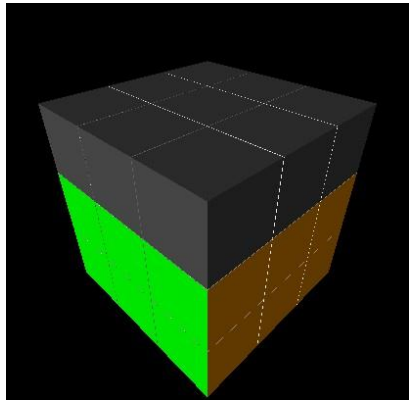
2.4 尋解 – 找出其解法

- stage 1: cross
  - 目標為完成底層十字
  - 每個邊有 24 種狀況，有 4 個邊
  - algorithm:
    - 一次完成一個邊(1,0,2)
    - 做一次 “y”，重複 4 次即可完成十字



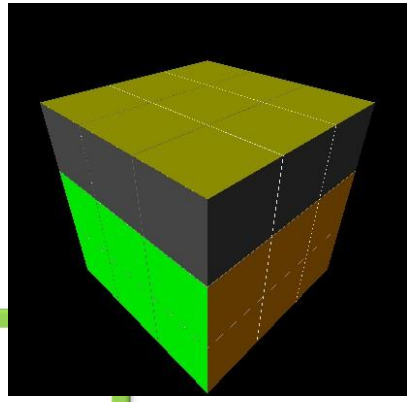
2.4 尋解 – 找出其解法

- stage 2: F2L (First 2 Layer)
  - 目標為完成下面兩層
  - 每個 pair 有 41 種狀況，有 4 個 pair
  - algorithm:
    - 預處理邊角位置
    - 套用 F2L 第一層公式(組合 pair)
    - 套用 F2L 第二層公式(pair 歸位)
    - 做一次 “y” 並重複前面步驟 4 次



2.4 尋解 – 找出其解法

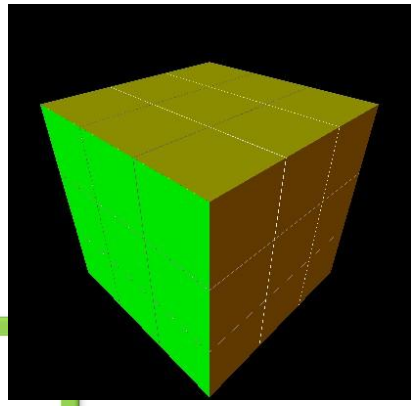
- stage 3: OLL (Orientation of Last Layer)
  - 頂層的方向，目標為讓頂層顏色正確
  - 一共有 57 種狀況
  - algorithm:
    - 把頂層的顏色編碼，尋找對應的公式



```
'''OLL_F[pattern] = moves
pattern format:
      0  1  2
   6  -  -  -  3
   7  -  -  -  4
   8  -  -  -  5
      9 10 11
len(OLL_F) = 58
done pattern = '000_000_000_000'
'''
OLL_F = {
    # pattern: moves
    '000_000_000_000': "",                          # done
    '010_111_111_010': "RU2R2FRF'U2R'FRF'",         # 01
    '011_010_111_011': "FRUR'U'F'fRUR'U'f'",
    '110_110_010_011': "L'R2BR'BLU2L'BLR'",
    '011_010_110_110': "RL2B'LB'R'U2RB'R'L",
    '110_100_011_000': "r'U2RUR'ULx",               # 05
```

2.4 尋解 – 找出其解法

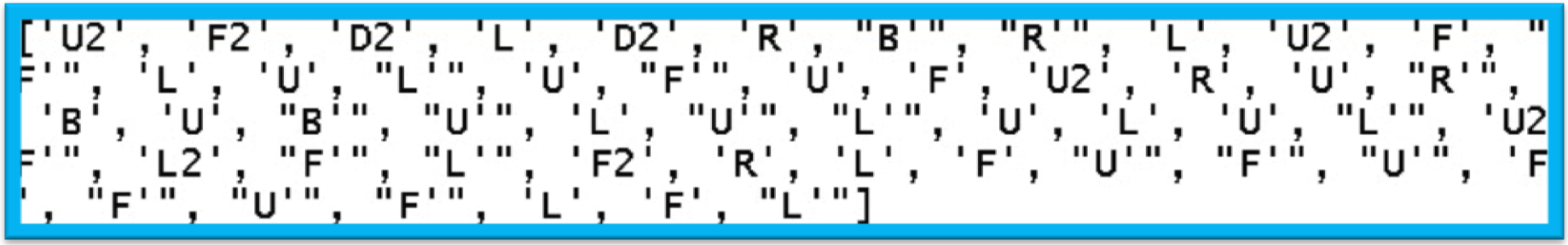
- stage 4: PLL (Permutation of Last Layer)
  - 頂層的排列，目標為完成整顆方塊
  - 一共有 21 種狀況
  - algorithm:
    - 把頂層的方塊位置編碼，尋找對應的公式



```
'''PLL_F[pattern] = moves
pattern format:
    012
    345
    678
len(PLL_F) = 22
done pattern = '012345678'
'''
PLL_F = {
    # pattern: moves
    '012345678': "",                                  # done
    '012547638': "R2URUR'U'R'U'R'UR'",                # P1
    '012743658': "RU'RURURU'R'U'R2",
    '016345872': "RB'RF2R'BRF2R2",
    '812345076': "L'BL'F2LB'L'F2L2",
    '032147658': "UR'U'RU'RURU'R'URUR2U'R'U",         # P5
    '072543618': "M2U'M2U2M2U'M2",
    '210345876': "xUR'U'LURU'L2x2U'RULU'R'U",
```

2.4 尋解 – 找出其解法

- 重整轉法
  - 把 “2”、”’” 全部展開
  - 把 fbudrl/MES extension，全部改用 FBUDRL 及 xyz 表示
  - 消除 xyz (用一張 table 維護舊轉法與新轉法的對應關系)
  - 重新合併相同轉法
- len(整理完的轉法) = 實際的步數

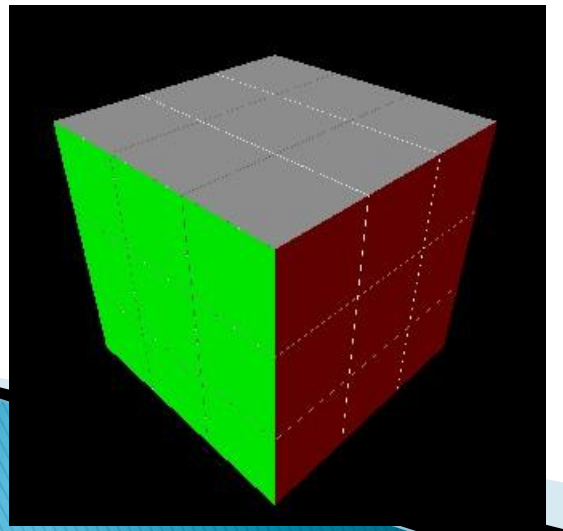


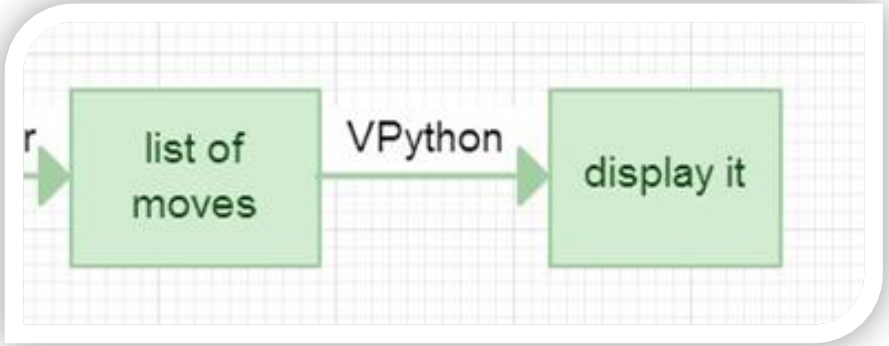
2.5 顯示 – 畫出一顆魔方

- VPython 是一個可以畫 3D 圖形的模組
- 每個魔方有 27 個小方塊
- 每個小方塊有 6 面需要畫
  - 畫 27 * 6 個 box

VPython
3D Programming for Ordinary Mortals

```
for x,y,z in itertools.product((0,1,2), repeat=3):
    sub_draw(self.pieces[x,y,z], x=x, y=y, z=z)
```

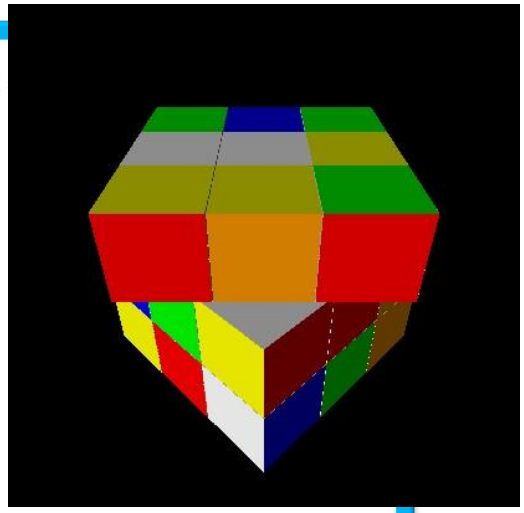




2.5 顯示 – 畫出一顆魔方

- 旋轉的動畫
  - 用 frame 把所有的 box 收集起來，再轉這個 frame

```
def anime(self, pieces, axis, twice, reverse)

    frame = visual.frame()
    for piece in pieces.flat:
        piece.frame.frame = frame

    fn = 20      # frame number
    sec = 0.3    # an action time (s)

    angle = visual.pi/2/fn
    if twice: angle *= 2
    if reverse: angle *= -1
    for i in xrange(fn):
        frame.rotate(angle=angle, axis=axis, origin=(1,1,1))
        #visual.sleep(sec/fn)
        visual.rate(fn/sec)
```



2.5 顯示 – 畫出一顆魔方

- 遇到的困難
  - VPython 在今年 2/19 出了第 6 版
    - 以前的 code 不會動了orz....
  - 陣列索引一回事；空間座標系一回事
    - 第一版和第二版的座標系不同XD

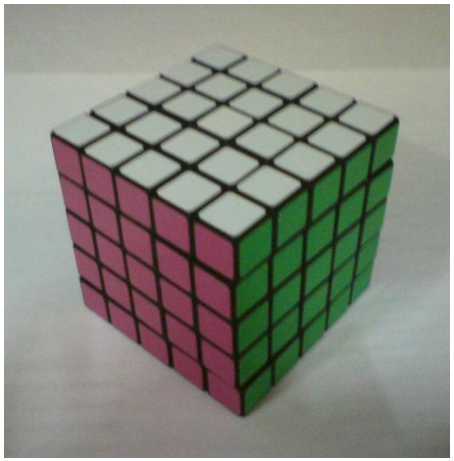
```
self.pieces = np.array([
    [
        [Piece( 0), Piece( 9), Piece(18)],
        [Piece( 3), Piece(12), Piece(21)],
        [Piece( 6), Piece(15), Piece(24)],
    ],

    [
        [Piece( 1), Piece(10), Piece(19)],
        [Piece( 4), Piece(13), Piece(22)],
        [Piece( 7), Piece(16), Piece(25)],
    ],

    [
        [Piece( 2), Piece(11), Piece(20)],
        [Piece( 5), Piece(14), Piece(23)],
        [Piece( 8), Piece(17), Piece(26)],
    ],
])
```

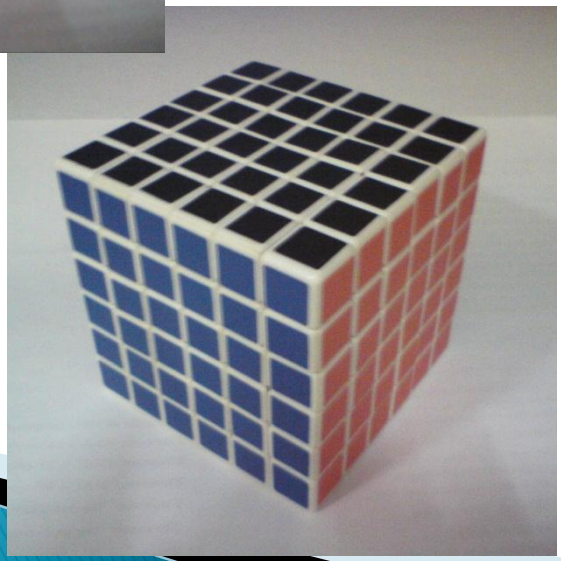
2.6 demo !

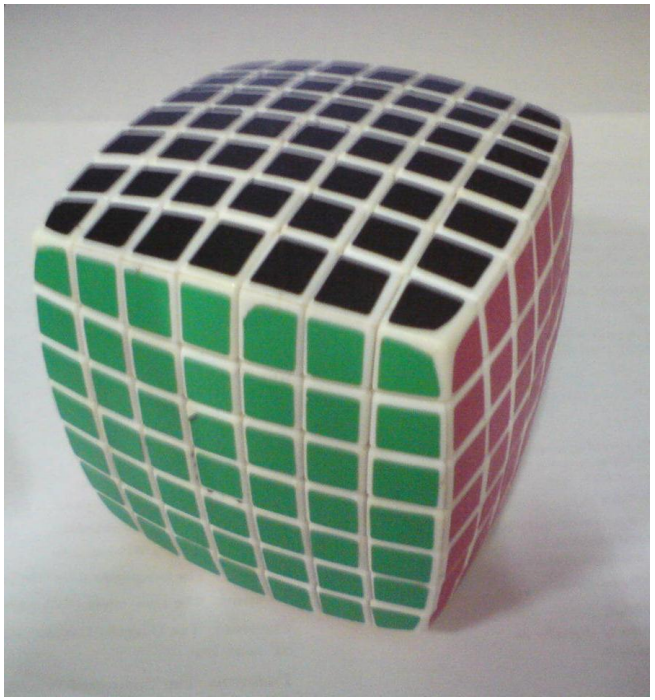
- Never live demo ?

2.7 pyRubiks 其他延伸應用

- 實作出其他種類的方塊

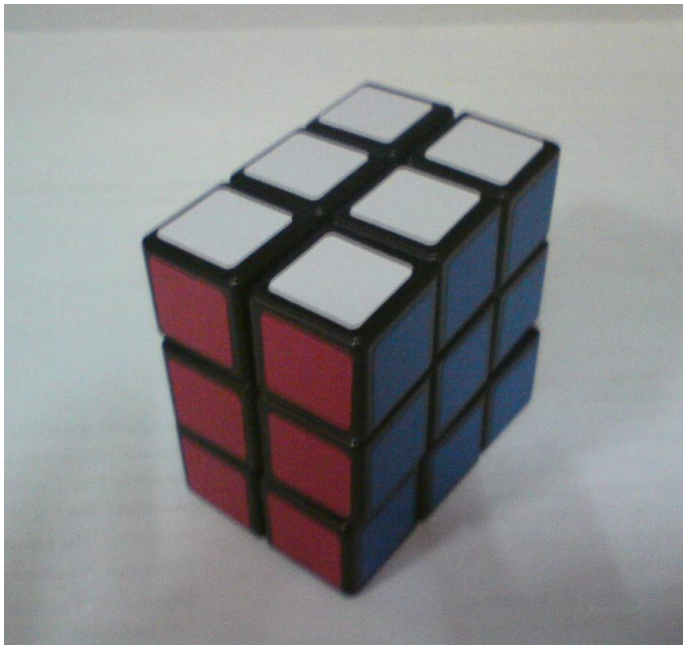






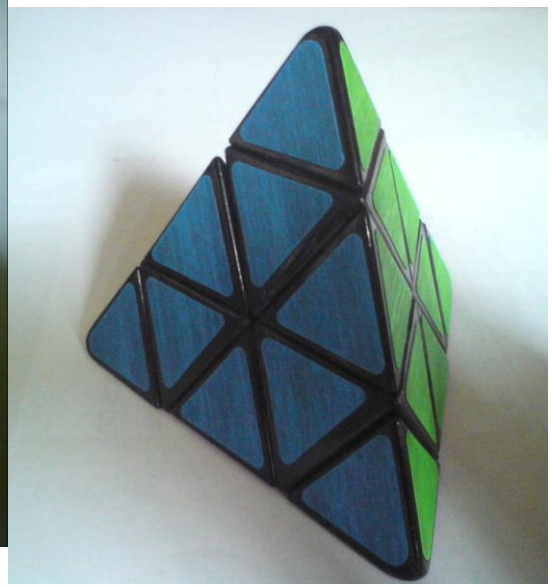
2.7 pyRubiks 其他延伸應用

- 實作出其他種類的方塊



2.7 pyRubiks 其他延伸應用

- 實作出其他種類的方塊



2.7 pyRubiks 其他延伸應用

- 實作出其他種類的方塊

2.7 pyRubiks 其他延伸應用

- 實作出其他種類的方塊

2.7 pyRubiks 其他延伸應用

- 與機器人連結，自動轉魔方機

3. 一些魔方記錄及軼事

類型	時間(單次最佳)	記錄保持人	我的記錄(平均)
2x2x2	0:00.69	Christian Kaserer	17 sec.
3x3x3	0:05.55	Mats Valk	35 sec.
4x4x4	0:26.44	Sebastian Weyer	4 min.
5x5x5	0:51.09	Feliks Zemdegs	6 min.
6x6x6	1:49.46	Kevin Hays	15 min.
7x7x7	2:41.63	Lin Chen	30 min.
3x3x3盲解	0:26.36	Marcell Endrey	∞
3x3x3單手	0:09.43	Giovanni Contardi	3 min.
3x3x3腳轉	0:27.93	Fakhri Raihaan	10+ min.

3. 一些魔方記錄及軼事

- 各種特別的轉魔方方式
  - 用腳轉 (**比賽項目**)
  - 用筷子轉
  - 邊丟邊轉

魔方世界歡迎您!

- Thank you for listening.