# T.I.P.E: Rubik's Cube 2x2x2

## I) Définitions des constantes et opérations de bases.

Les constantes sont en principe écrites tout en majuscule, à l'exception des plus usitées qui sont alors tout en minuscule.

*On charge tout d'abord le module utilisé:

$with(plots):$

### A) Les couleurs

*Tableau (C) des 6 couleurs utilisées dans le cube (version officielle):
Ce tableau sert essentiellement à obtenir les couleurs de chaque face pour les options des plots, la manipulation pour les mouvements du Rubik's Cube se faisant alors au niveau des indices, on a donc Chifre⇒Couleur(en Maple)

$C[1] := red:$
$C[2] :=$ "DarkGreen" :
$C[3] :=$ "OrangeRed" :
$C[4] := blue:$
$C[5] := yellow:$
$C[6] := white:$

*Attributions des couleurs à chaque indice de C (initiales anglaises pour éviter les confusions bleu/blanc) :
Ces notations permettent exclusivement de facilité l'entrée des couleurs de chaque face d'un Rubik's Cube donné, en pratique on s'en sert de la manière suivante: Couleur(en "francais")⇒Chifre⇒Indice⇒Couleur(en Maple)

$r := 1:$
$g := 2:$
$o := 3:$
$b := 4:$
$y := 5:$
$w := 6:$

### B) Le Rubik's Cube

Le Rubik's Cube est un tableau composé par chacun de ses 8 petits cubes.

Les petits cubes sont eux même des tableaux composé de leurs coordonnées et des couleurs de leurs 3 faces.
Les coordonnees sont eux aussi des tableaux donnée par 3 chiffres (d'après les notations adopté).
Les couleurs sont finallement toujours des tableaux donnée par les 3 couleurs des faces du petit cube concerné.

<u>*Définitions des 8 cubes d'un Rubik's Cube fini:</u>
Le numéro de chaque cube n'a aucune importance, ils sont prit dans l'ordre de la notation choisit.

$$CUBE.FINI[1] := \left[ \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}, \begin{bmatrix} C[w] \\ C[r] \\ C[b] \end{bmatrix} \right]:$$

$$CUBE.FINI[2] := \left[ \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} C[g] \\ C[r] \\ C[w] \end{bmatrix} \right]:$$

$$CUBE.FINI[3] := \left[ \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix}, \begin{bmatrix} C[b] \\ C[r] \\ C[y] \end{bmatrix} \right]:$$

$$CUBE.FINI[4] := \left[ \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix}, \begin{bmatrix} C[y] \\ C[r] \\ C[g] \end{bmatrix} \right]:$$

$$CUBE.FINI[5] := \left[ \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} C[b] \\ C[o] \\ C[w] \end{bmatrix} \right]:$$

$$CUBE.FINI[6] := \left[ \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}, \begin{bmatrix} C[w] \\ C[o] \\ C[g] \end{bmatrix} \right]:$$

$$CUBE.FINI[7] := \left[ \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}, \begin{bmatrix} C[y] \\ C[o] \\ C[b] \end{bmatrix} \right]:$$

$$CUBE.FINI[8] := \left[ \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}, \begin{bmatrix} C[g] \\ C[o] \\ C[y] \end{bmatrix} \right]:$$

<u>*Notations du premier tableau d'un cube (les coordonnées) et du second (les couleurs):</u>

$coor := 1 :$
$coul := 2 :$

*Définitions du Rubik's Cube fini:

$RUBIX.FINI := [CUBE.FINI[1], CUBE.FINI[2], CUBE.FINI[3], CUBE.FINI[4], CUBE.FINI[5], CUBE.FINI[6], CUBE.FINI[7], CUBE.FINI[8]]:$

(Compte tennu de l'utilisation ultra fréquente de ce dernier dans les arguments des autres fonctions, on prendra la notation $rubix := RUBIX.FINI:$ jusqu'à l'aboutissement du programme)

*Fonction de correspondance n° Petit Cube ⇒ Coordonnée:
C'est une fonction qui donne les coordonnées corespondantes à un petit cube désigner par son numéro dans la notation choisit.

```
NumeroVersCoordonnees :=proc (num,$)
                    local x, y, z;
#La première coordonnée se récupère très facilement
                    x := 0^(num mod 2);
#Les deux autres nécessite une boucle conditionnelle
                    if num = 1 then y := 0; z := 1;
                  elif num = 2 then y := 0; z := 0;
                  elif num = 3 then y := 1; z := 1;
                  elif num = 4 then y := 1; z := 0;
                  elif num = 5 then y := 0; z := 0;
                  elif num = 6 then y := 0; z := 1;
                  elif num = 7 then y := 1; z := 0;
                  elif num = 8 then y := 1; z := 1;
                    fi;
#On renvoie finnallement le résultat
                  [x, y, z];
                  end proc:
```

*Fonction de correspondance Coordonnée ⇒n° Peti Cube:
C'est une fonction qui donne le numéro d'un petit cube dans la notation choisit en fonction de ses coordonnées .

```
CoordonneesVersNumero :=proc (x, y, z,$)
 #Une boucle conditionnelle est la façon la plus claire pour avoir ce qu'on veut
                    if (x, y, z) = (0, 0, 1)  then 1
                  elif (x, y, z) = (1, 0, 0)  then 2
                  elif (x, y, z) = (0, 1, 1)  then 3
                  elif (x, y, z) = (1, 1, 0)  then 4
                  elif (x, y, z) = (0, 0, 0)  then 5
                  elif (x, y, z) = (1, 0, 1)  then 6
                  elif (x, y, z) = (0, 1, 0)  then 7
                  elif (x, y, z) = (1, 1, 1)  then 8
                   fi
          end proc:
```

*Fonction de rafraîchissement du Rubik's cube, qui le place dans sa notation standard:

```
Rafraichir :=proc(Rubix1,$)
          local k, Rubix2;
```

#On replace les positions de chaque petit cube

```
        for k to 8 do
        Rubix2[CoordonneesVersNumero(Rubix1[k][coor][1], Rubix1[k][coor][2],
    Rubix1[k][coor][3])] := Rubix1[k]
        end do;
```

#On renvoie le nouveau Rubik's Cube

```
        Rubix2;
        end proc:
```

# C) Les etats du Rubik's Cube

## α) La finitude

*Fonction testant si le Rubik's Cube considéré est fini:
On vérifie simplement que les 4 petites faces de chacune des 6 grandes faces sont de la même couleur.

```
EstFini :=proc(RubixTeste,$)
    local k, x, y, z, Rubix;
#On renumérote les petits cube du Rubik's Cube pour le placer dans la position de références
    for k to 8 do
    #On récupère les coordonées de chaque petit cube
    x := RubixTeste[k][coor][1];
    y := RubixTeste[k][coot][2];
    z := RubixTeste[k][coor][3];

  #Une formulle barbare renumérote chacun des petits cubes du Rubik's Cube selon leurs
  coordonnées
    #pour les faire correspondre avec la notation choisit.
    Rubix[0^x·(0^y·(5 − 4 ·z) + y·(7 − 4·z)) + x·(0^y·(2 + 4·z) + y·(4 + 4·z))]
  := RubixTeste[k];
    end do ;
#Un gros booléens donne alors directement la réponse
     Rubix[1][coul][2] = Rubix[2][coul][2] = Rubix[3][coul][2] = Rubix[4][coul][2]
 and Rubix[5][coul][2] = Rubix[6][coul][2] = Rubix[7][coul][2] = Rubix[8][coul][2]
 and Rubix[2][coul][1] = Rubix[4][coul][3] = Rubix[6][coul][3] = Rubix[8][coul][1]
 and Rubix[1][coul][3] = Rubix[3][coul][1] = Rubix[5][coul][1] = Rubix[7][coul][3]
 and Rubix[1][coul][1] = Rubix[2][coul][3] = Rubix[5][coul][3] = Rubix[6][coul][1]
 and Rubix[3][coul][3] = Rubix[4][coul][1] = Rubix[7][coul][1] = Rubix[8][coul][3]
     end proc:
```

## β) La faisabilité

Comme il existe une suite de mouvement permettant de interchanger 2 petits cubes quelconques sans changer leur orientation, on ne considère que les orientations des petits cubes.
Il est alors nécesaire et suffisant d'avoir:
• $\forall$Cube1,Cube2$\in$Rubix,{Cube1[coul]}$\neq${Cube2[coul]}
• $\forall$Cube$\in$Rubix,$\exists$CUBE.FINI$\in$RUBIX.FINI |$\exists$n$\in\mathbb{N}$| Cube[coul]=$c^n(CUBE.FINI[coul])$ : où c(E) est un cycle de l'ensemble E
•$\Sigma$n ≡ 0 [3]

*Axiome 1:

Chaque petits cubes d'un Rubik's Cube faisable doivent necessairement avoir leur triplets de couleur différent entre eux.

```
Axiome1 := proc(Rubix,$)
 #On vérifie qu'il y ait bien 8 triplets différents de couleurs
        evalb(nops( {
                {Rubix[1][coul][1], Rubix[1][coul][2], Rubix[1][coul][3]},
                {Rubix[2][coul][1], Rubix[2][coul][2], Rubix[2][coul][3]},
                {Rubix[3][coul][1], Rubix[3][coul][2], Rubix[3][coul][3]},
                {Rubix[4][coul][1], Rubix[4][coul][2], Rubix[4][coul][3]},
                {Rubix[5][coul][1], Rubix[5][coul][2], Rubix[5][coul][3]},
                {Rubix[6][coul][1], Rubix[6][coul][2], Rubix[6][coul][3]},
                {Rubix[7][coul][1], Rubix[7][coul][2], Rubix[7][coul][3]},
                {Rubix[8][coul][1], Rubix[8][coul][2], Rubix[8][coul][3]},
        }) = 8);
        end proc:
```

*Fonction qui vérifie si un triplet de couleur est cycle du rubix fini:

```
EstCycle := proc(Couleur,$)
        local k, n;
        n := ∞;
#On vérifie que le vecteur de couleur donnée est cycle du rubix fini
        for k to 8 do
        if Couleur[1] = (RUBIX.FINI)[k][coul][1] and Couleur[2] = (RUBIX.FINI)[k][coul][2]
    and Couleur[3] = (RUBIX.FINI)[k][coul][3] then n := 0; break;
      elif Couleur[1] = (RUBIX.FINI)[k][coul][2] and Couleur[2] = (RUBIX.FINI)[k][coul][3]
    and Couleur[3] = (RUBIX.FINI)[k][coul][1] then n := 1; break;
      elif Couleur[1] = (RUBIX.FINI)[k][coul][3] and Couleur[2] = (RUBIX.FINI)[k][coul][1]
    and Couleur[3] = (RUBIX.FINI)[k][coul][2] then n := 2; break;
        end if; end do;
#On renvoie l'indice du cycle
        n;
        end proc:
```

*Axiome 2-3:

L'orientation de chaque petits cubes d'un Rubik's Cube faisable doit necessairement être un cycle d'un des petits petits cube du cube fini.

```
Axiome2et3 := proc(Rubix,$)
        local n, k;
        n := 0;
#On récupère l'indice des cycle de chaque cube
        for k to 8 do
        n := n + EstCycle(Rubix[k][coul]);
        end do;
 #On vérifie d'un coup les deux axiomes
        (not n = ∞) and (n mod 3 = 0);
        end proc:
```

# II)Les 12 mouvements possibles

## A)Selon X

*Fonction qui change les données d'un petit cube ayant subi une rotation du bloc X0 (dans le sens direct) :

$RotationCubeX0 := \mathbf{proc}(Cube,\$)$
$\mathbf{local}\, x, y, z, C, a, b, c, d, e, f;$
*#On récupère les coordonnées du petit cube donné*
$x := Cube[coor][1];$
$y := Cube[coor][2];$
$z := Cube[coor][3];$
*#On récupère l'orientation des couleurs du petit cube donné*
$C[0] := Cube[coul][1];$
$C[1] := Cube[coul][2];$
$C[2] := Cube[coul][3];$
*#On détermine les nouvelles coordonnées du petit cube*
$a := x;$
$b := x \cdot y + (1 - x) \cdot \left(0^{z + y + 1 \mathbf{\,mod\,} 2} + y\right) \mathbf{\,mod\,} 2;$
$c := x \cdot z + (1 - x) \cdot \left(0^{(z + y) \mathbf{\,mod\,} 2} + z\right) \mathbf{\,mod\,} 2;$
*#On détermine les nouvelles orientations des couleurs du petit cube*
$d := \left( x \cdot 0 + (1 - x) \cdot (y + z \mathbf{\,mod\,} 2) \cdot 2 + (1 - x) \cdot 0^{y + z \mathbf{\,mod\,} 2} \cdot 1 \right) \mathbf{\,mod\,} 3;$
$e := \left( x \cdot 1 + (1 - x) \cdot (y + z \mathbf{\,mod\,} 2) \cdot 0 + (1 - x) \cdot 0^{y + z \mathbf{\,mod\,} 2} \cdot 2 \right) \mathbf{\,mod\,} 3;$
$f := \left( x \cdot 2 + (1 - x) \cdot (y + z \mathbf{\,mod\,} 2) \cdot 1 + (1 - x) \cdot 0^{y + z \mathbf{\,mod\,} 2} \cdot 0 \right) \mathbf{\,mod\,} 3;$
*#On renvoie les nouvelles données du cube*

$$\left[\begin{bmatrix} a \\ b \\ c \end{bmatrix}, \begin{bmatrix} C[d] \\ C[e] \\ C[f] \end{bmatrix}\right];$$

**end proc**:

*Fonction qui applique celle qui précède à tous les petit cube du rubik's cube et renvoie le nouveau:

$RotationBlocX0 := \mathbf{proc}(Rubix1,\$)$
$\mathbf{local}\, k, Rubix2;$
*#On change tous les petits cube du Rubik's considéré*
**for** $k$ **to** 8 **do**
$Rubix2[k] := RotationCubeX0(RotationCubeX0(RotationCubeX0(Rubix1[k])));$
**end do**;
$Rubix2;$
**end proc**:

*Fonction qui fait cette fois ci l'opération dans le sens indirect:

$RotationBlocX0inv := \mathbf{proc}(Rubix1,\$)$
$\mathbf{local}\, k, Rubix2;$
*#On change tous les petits cube du Rubik's considéré*

```
 for k to 8 do
Rubix2[k] := RotationCubeX0(Rubix1[k]);
end do;
Rubix2;
end proc:
```

Compte tennue de l'utilisation plus que récurente de ces deux fonctions pour la saisi des mouvement, on les nommes égallement, jusqu'à la fin du projet, de manière moins explicite mais plus simple par $x0 := RotationBlocX0:$ et $x0'' := RotationBlocX0inv:$

*Fonction qui change les données d'un petit cube ayant subi une rotation du bloc X1 (dans le sens direct) :*

```
RotationCubeX1 := proc(Cube,$)
            local x, y, z, C, a, b , c, d, e, f;
#On récupère les coordonnées du petit cube donné
            x := Cube[coor][1];
            y := Cube[coor][2];
            z := Cube[coor][3];
#On récupère l'orientation des couleurs du petit cube donné
            C[0] := Cube[coul][1];
            C[1] := Cube[coul][2];
            C[2] := Cube[coul][3];
#On détermine les nouvelles coordonnées du petit cube
            a := x;
```

$b := (1-x)\cdot y + x\cdot\left(0^{z+y+1 \bmod 2} + y\right) \bmod 2;$

$c := (1-x)\cdot z + x\cdot\left(0^{(z+y) \bmod 2} + z\right) \bmod 2;$

```
#On détermine les nouvelles orientations des couleurs du petit cube
```

$d := \left((1-x)\cdot 0 + x\cdot(y+z \bmod 2)\cdot 2 + x\cdot 0^{y+z \bmod 2}\cdot 1\right) \bmod 3;$

$e := \left((1-x)\cdot 1 + x\cdot(y+z \bmod 2)\cdot 0 + x\cdot 0^{y+z \bmod 2}\cdot 2\right) \bmod 3;$

$f := \left((1-x)\cdot 2 + x\cdot(y+z \bmod 2)\cdot 1 + x\cdot 0^{y+z \bmod 2}\cdot 0\right) \bmod 3;$

```
#On renvoie les nouvelles données du cube
```

$$\left[\begin{bmatrix} a \\ b \\ c \end{bmatrix}, \begin{bmatrix} C[d] \\ C[e] \\ C[f] \end{bmatrix}\right];$$

```
           end proc:
```

*Fonction qui applique celle qui précède à tous les petit cube du rubik's cube et renvoie le nouveau:*

```
RotationBlocX1 :=proc(Rubix1,$)
           local k, Rubix2;
 #On change tous les petits cube du Rubik's considéré
           for k to 8 do
           Rubix2[k] := RotationCubeX1(Rubix1[k]);
           end do;
           Rubix2;
           end proc:
```

*Fonction qui fait cette fois ci l'opération dans le sens indirect:*

```
RotationBlocX1inv := proc(Rubix1,$)
                    local k, Rubix2;
 #On change tous les petits cube du Rubik's considéré
                    for k to 8 do
                   Rubix2[k] := RotationCubeX1(RotationCubeX1(RotationCubeX1(Rubix1[k])));
                   end do;
                   Rubix2;
                   end proc:
```

Compte tennue de l'utilisation plus que récurente de ces deux fonctions pour la saisi des mouvement, on les nommes égallement, jusqu'à la fin du projet, de manière moins explicite mais plus simple par
*x1* := *RotationBlocX1* : et *x1"* := *RotationBlocX1inv* :

# B)Selon Y

*Fonction qui change les données d'un petit cube ayant subi une rotation du bloc Y0 (dans le sens direct) :

```
RotationCubeY0 := proc(Cube,$)
                    local x, y, z, C, a, b , c, d, e, f;
#On récupère les coordonnées du petit cube donné
                    x := Cube[coor][1];
                    y := Cube[coor][2];
                    z := Cube[coor][3];
#On récupère l'orientation des couleurs du petit cube donné
                   C[0] := Cube[coul][1];
                   C[1] := Cube[coul][2];
                   C[2] := Cube[coul][3];
#On détermine les nouvelles coordonnées du petit cube
                   a := y·x + (1 − y)·(x + 0^z) mod 2;
                   b := y;
                   c := y·z + (1 − y)·(x + z + 0^x) mod 2;
#On détermine les nouvelles orientations des couleurs du petit cube
                   d := ( y·0 + (1 − y)·z·1 + (1 − y)·(1 − z)·2) mod 3;
                   e := (y·1 + (1- y)·z·2 + (1 − y)·(1 − z)·0) mod 3;
                   f := (y·2 + (1 − y)·z·0 + (1 − y)·(1 − z)·1) mod 3;
#On renvoie les nouvelles données du cube
                   [[a, b, c], [C[d], C[e], C[f]]];
                 end proc:
```

*Fonction qui applique celle qui précède à tous les petit cube du rubik's cube et renvoie le nouveau: