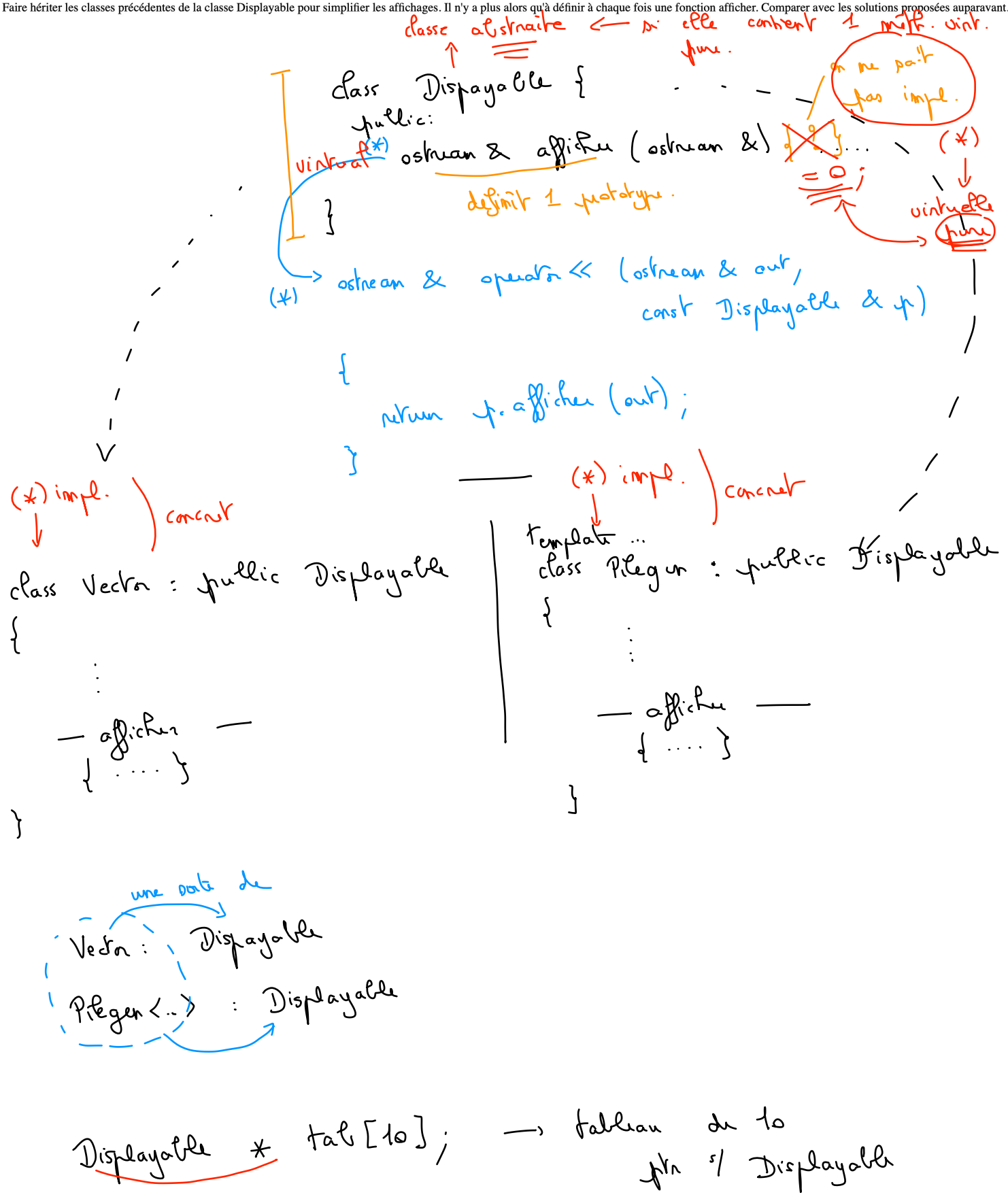
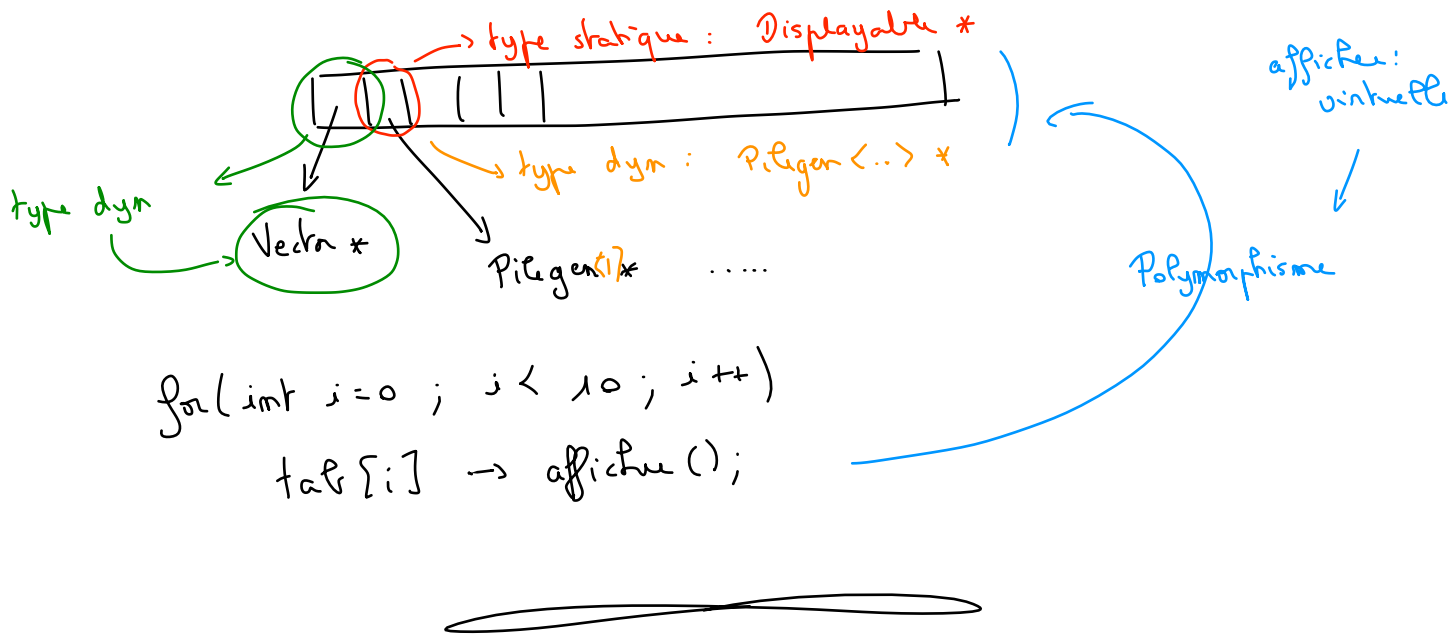
****
Partie II : classe Displayable *
****

Introduire une classe abstraite Displayable permettant de gérer la redéfinition de l'opérateur << par la redéfinition d'une fonction membre virtuelle pure afficher.
La redéfinition de l'opérateur sera réalisée pour s'appliquer sur un objet p :
const Displayable & p ou
const Displayable * p

Faire hériter les classes précédentes de la classe Displayable pour simplifier les affichages. Il n'y a plus alors qu'à définir à chaque fois une fonction afficher. Comparer avec les solutions proposées auparavant.






```
for(int i=0 ; i < 10 ; i++)
    tab[i] -> affiche();
```

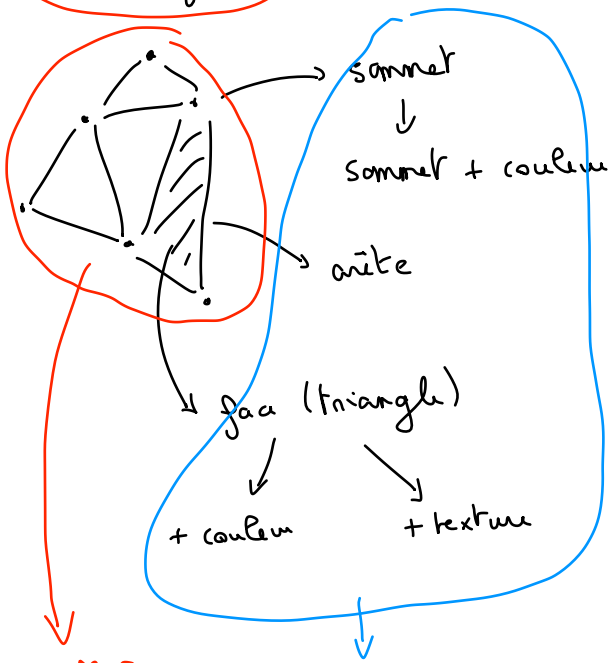
Ex :

Displayable

```
virtual ... affiche ... = 0;
```

3D – OpenGL → maillages

moteur d'affichage



classes :

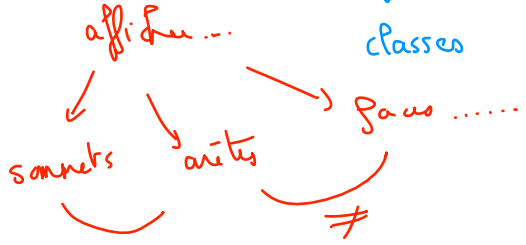
Sommet (3 coords) → affiche (OpenGL)

Sommet_color : Sommet → affiche

Face → 3 sommets.... → affiche

Face_color : Face → affiche

⋮



Moteur graphique :

vector ....

```
Displayable * moteur[100];
```

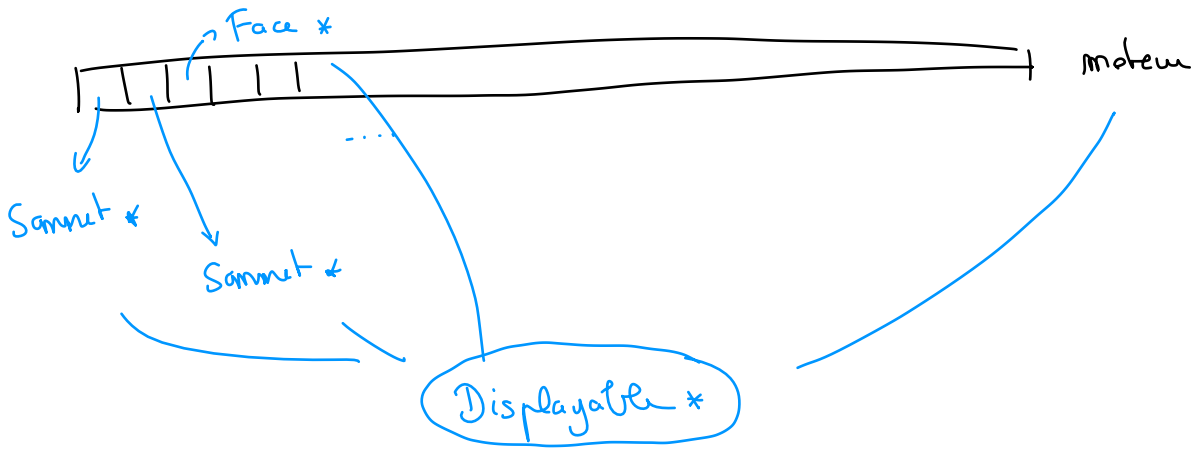
↓

```
for(i=0; i<100; i++)
    moteur[i] -> affiche();
```

polymorphisme ...

→ vecteur de sommet / face / arête / .... / ....

→ on les parcourt, on affiche



1. Si on définit :
int i = 0 ;
int &j ;
Comment peut-on initialiser j?

j = &i ;	(2) 25%
j = *i ;	(2) 25%
j=i ;	(4) 50%

int & j ≠ &i : adresse de i → int *

int * p = &i ;

i : int

~~*i~~ → déréférence 1 ptrs

(*p)

int i = 0 ;
int & j ;
j = i ;

→ ds j on met 1 réf s/ i

2. Dans la classe Toto, on définit :
int ma_methode (const Vector & v) const ;
Le premier const signifie que:

La méthode ne change pas l'objet dans lequel on est	(1) 13%
✓ La méthode ne change pas l'argument v	(7) 88%

const Vector & v → on passe 1 réf s/ v → on garantit qu'on ne modifiera pas v

3. Dans la même classe, le second const signifie que:

✓ La méthode ne change pas l'objet dans lequel on est	(7) 88%
La méthode ne change aucun argument	(1) 13%

→ la méthode est const

ne modifie pas this.

const

4. Si dans la fonction ma_methode, j'appelle la méthode
this->afficher() ;
Alors afficher doit être déclarée comme :

void afficher() ; (3) 38%

void afficher() const ; (5) 63%

int ma_methode(const Vecto &v) const

{

this -> adjust_size(10);

this -> afficher(); ⇒ const

}

la meth. ne mod. pas this

tout ce qu'on fait dedans doit le garantir.

⚠ toute méthode const l'est déclarée systématiquement

```
class Tote {
  private:
    int _i;
  public:
    int meth1() const { return _i; }
    void meth2(int i) { _i = i; }
};
```

get....

ne pas const.

set ...

Pilegen

- → push
- → pop
- → top
- → est_vide

- → afficher
- → taille (getter...)

Pilegen < Pilegen < Vector > > f ;

C++

int ~> classes
float ~>

int i(3) ;

constructeur prenant 1 entier.

~~Pilegen~~ — n'est pas 1 classe - type

Pilegen < ... > — classe / type.
