tikz3d-fr

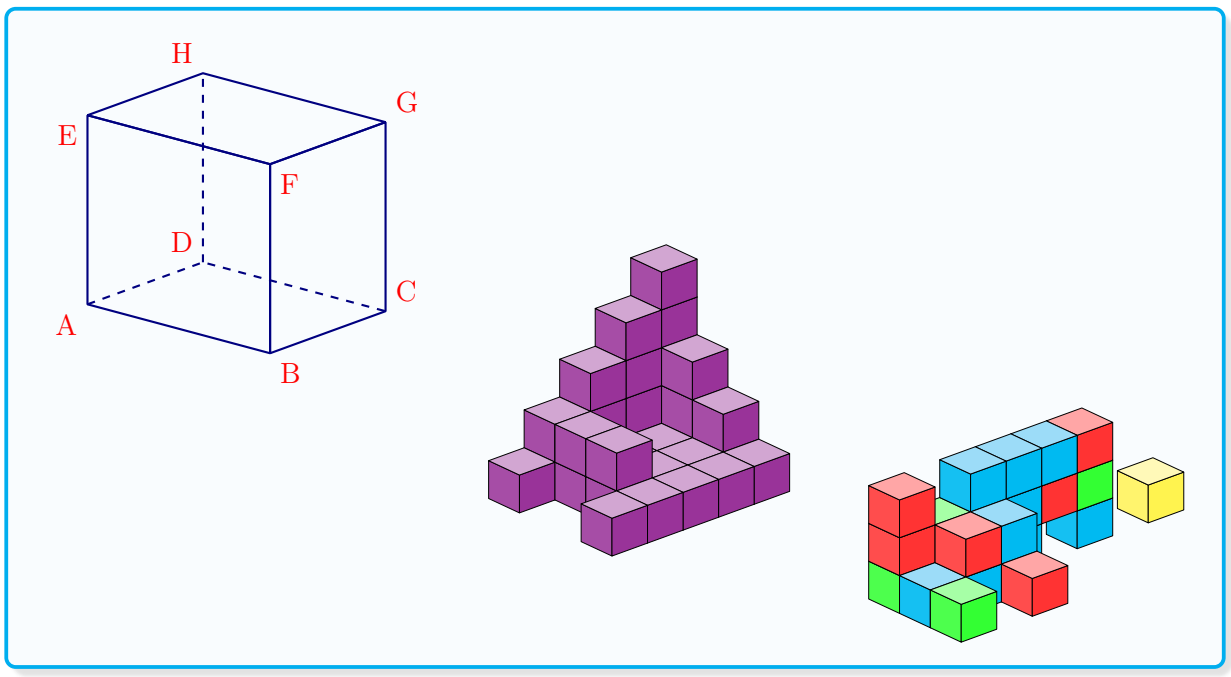
Quelques commandes (fr) pour

un peu de 3D avec Ti*k*Z.

Version 0.1.2 - 22 juillet 2023

Cédric Pierquet
c pierquet - at - outlook . fr
https://github.com/cpierquet/tikz3dfr

- Un environnement avec déclaration des unités.
- Une commande pour afficher un pavé, avec personnalisations.
- Deux commandes pour afficher des empilements de « petits cubes ».



LaTeX

pdfLaTeX

LuaLaTeX

Ti*k*Z

TeXLive

MiKTeX

Table des matières

I Historique **2**

II Introduction **3**

1 Le package tikz3d-fr **3**
1.1 Introduction 3
1.2 Chargement du package, packages utilisés 3

III Environnement 3D et commandes de base **4**

2 Création de l'environnement **4**
2.1 Commande 4
2.2 Clés et options 4

3 Points et segments **6**
3.1 Commandes 6
3.2 Clés, options et arguments 6

4 Pavés **8**
4.1 Commandes 8
4.2 Clés et options 8

IV Empilements de cubes **10**

5 Environnement dédié **10**

6 Création par *plaques* **10**
6.1 Commandes 10
6.2 Options et arguments 11

7 Création par *hauteurs* **13**
7.1 Commande 13
7.2 Options et arguments 13

Première partie
Historique

`v0.1.2` : Option pour l'épaisseur des traits + [line join=round] pour les cubes
`v0.1.1` : Tracé des segments individuels avec l'option [line cap=round]
`v0.1.0` : Version initiale

Deuxième partie

Introduction

1 Le package tikz3d-fr

1.1 Introduction

Le package propose des commandes basiques – et francisées – pour travailler sur des figures simples en 3D, à l'aide de Ti*k*Z en utilisant des coordonnées tridimensionnelles :

- un environnement avec gestion des unités $x/y/z$;
- une commande pour tracer et personnaliser un cube ;
- des commandes pour créer/afficher/nommer des points de l'espace ;
- des commandes pour tracer un ou plusieurs segments ;
- des commandes et un environnement pour travailler sur des *empilements* de cubes.

Il existe d'autres solutions pour travailler avec de la 3D en LaTeX, comme par exemple les packages ProfCollege [1] (de Christophe Poulain, qui utilise MetaPost, et qui est certainement beaucoup plus performant) ou pst-ob3d [2] (de Herbert Voß et Denis Girou, qui utilise PSTricks).

L'idée est de proposer une utilisation des capacités (natives) 3D de Ti*k*Z, en proposant des commandes *simplifiées* et *francisées* pour des figures simples (tétraèdres, cubes, pyramides, pavés) utilisées fréquemment dans des exercices de géométrie dans l'espace dans l'enseignement secondaires en France.

1.2 Chargement du package, packages utilisés

Le package se charge, de manière classique, dans le préambule.
Il n'existe pas d'option pour le package, et `xcolor` n'est pas chargé.

Code LaTeX

```
\usepackage{tikz3d-fr}
```

tikz3d-fr charge les packages suivantes :

- `tikz`, `xstring`, `xintexpr`, `simplekv` et `xinttools` et `listofitems` ;
- les librairies *`tikz`*`.calc` et *`tikz`*`.babel`.

Il est compatible avec les compilations usuelles en latex, pdflatex, lualatex ou xelatex.

1. https://www.ctan.org/pkg/profcollege
2. https://www.ctan.org/pkg/pst-ob3d

Troisième partie

Environnement 3D et commandes de base

2 Création de l'environnement

2.1 Commande

L'environnement dédié à la création de figures en 3D avec Ti*k*Z est `EnvTikzEspace`.
Il permet de définir les unités et *angles* des différents axes.

Code LaTeX

```
\begin{EnvTikzEspace}[Clés]<options tikz>
    %commandes
\end{EnvTikzEspace}
```

2.2 Clés et options

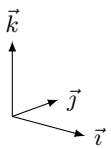
Le premier argument, optionnel et entre `[...]` propose les ⟨**clés**⟩ suivantes :

- ⟨**UniteX**⟩, à donner sous la forme ⟨**{angle:longueur}**⟩ pour le vecteur de base $\vec{\imath}$; défaut : ⟨**{-15:1cm}**⟩,
- ⟨**UniteY**⟩, à donner sous la forme ⟨**{angle:longueur}**⟩ pour le vecteur de base $\vec{\jmath}$; défaut : ⟨**{20:0.65cm}**⟩
- ⟨**UniteZ**⟩, à donner sous la forme ⟨**{angle:longueur}**⟩ pour le vecteur de base $\vec{k}$; défaut : ⟨**{90:1cm}**⟩
- Le booléen ⟨**VueClassique**⟩ pour un affichage en perspective habituelle. défaut : ⟨**false**⟩

Le second argument, optionnel et entre `<...>` est quant à lui relatif à des arguments à passer à l'environnement Ti*k*Z créé, comme par exemple un alignement vertical, etc

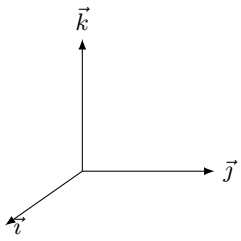
Code LaTeX et sortie LaTeX

```
\begin{EnvTikzEspace}
    \draw[->,>=latex] (0,0,0)--(1,0,0) node[right] {$\vec{\imath}$} ;
    \draw[->,>=latex] (0,0,0)--(0,1,0) node[right] {$\vec{\jmath}$} ;
    \draw[->,>=latex] (0,0,0)--(0,0,1) node[above] {$\vec{k}$} ;
\end{EnvTikzEspace}
```



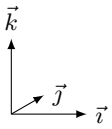
Code LaTeX et sortie LaTeX

```
\begin{EnvTikzEspace}[UniteX={-145:1.25cm},UniteY={0:1.75cm},UniteZ={90:1.75cm}]
    \draw[->,>=latex] (0,0,0)--(1,0,0) node[right] {$\vec{\imath}$} ;
    \draw[->,>=latex] (0,0,0)--(0,1,0) node[right] {$\vec{\jmath}$} ;
    \draw[->,>=latex] (0,0,0)--(0,0,1) node[above] {$\vec{k}$} ;
\end{EnvTikzEspace}
```



Code LaTeX et sortie LaTeX

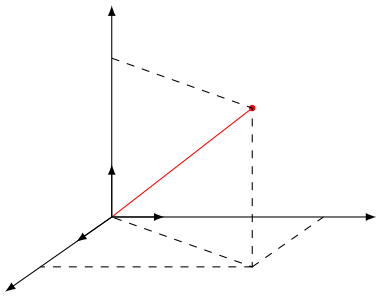
```
\begin{EnvTikzEspace}[VueClassique]
    \draw[->,>=latex] (0,0,0)--(1,0,0) node[right] {$\vec{\imath}$} ;
    \draw[->,>=latex] (0,0,0)--(0,1,0) node[right] {$\vec{\jmath}$} ;
    \draw[->,>=latex] (0,0,0)--(0,0,1) node[above] {$\vec{k}$} ;
\end{EnvTikzEspace}
```



Une fois l'environnement, via son *repère*, est créé, toute commande en Ti*k*Z est utilisable avec les coordonnées `(x,y,z)` ou bien les `scope` avec les `canva`.

Code LaTeX et sortie LaTeX

```
\begin{EnvTikzEspace}[UniteX={-145:0.575cm},UniteY={0:0.7cm},UniteZ={90:0.7cm}]
    \filldraw[red] (2,4,3) circle[radius=1pt] ; \draw[red] (0,0,0) -- (2,4,3) ;
    \draw[thin,->,>=latex] (0,0,0)--(3,0,0) ;
    \draw[thin,->,>=latex] (0,0,0)--(0,5,0) ;
    \draw[thin,->,>=latex] (0,0,0)--(0,0,4) ;
    \draw[->,>=latex] (0,0,0)--(1,0,0) ;
    \draw[->,>=latex] (0,0,0)--(0,1,0) ;
    \draw[->,>=latex] (0,0,0)--(0,0,1) ;
    \draw[dashed] (0,0,3)--(2,4,3)--(2,4,0)--(0,0,0) (2,4,0)--(2,0,0) (2,4,0)--(0,4,0) ;
\end{EnvTikzEspace}
```



3 Points et segments

3.1 Commandes

Les commandes *simplifiées* et *francisées* disponibles sont :

— `\PlacePointEspace` pour placer un point dans l'espace ;
— `\PlacePointsEspace` pour placer des points dans l'espace ;
— `\MarquePointEspace` pour marquer (matérialiser) un point dans l'espace ;
— `\MarquePointsEspace` pour marquer (matérialiser) un point dans l'espace ;
— `\TraceSegmentEspace` pour tracer un segment dans l'espace ;
— `\TraceSegmentsEspace` pour tracer des segments dans l'espace.

Code LaTeX

```
\begin{EnvTikzEspace}[Clés]<options tikz>
    %créer/placer/nommer un point
    \PlacePointEspace(*)[clés]{nœud}{coordonnées}<label>
    %créer/placer/nommer plusieurs points
    \PlacePointsEspace(*)[clés]{liste}
    %marquer un point
    \MarquePointEspace[clés]{point}
    %marquer plusieurs points
    \MarquePointsEspace[clés]{liste}
    %tracer un segment
    \TraceSegmentEspace[clés](point)(point)
    %tracer plusieurs segments
    \TraceSegmentsEspace[clés]{liste}
\end{EnvTikzEspace}
```

3.2 Clés, options et arguments

Les versions étoilées désactivent l'affichage des labels des points.
L'argument optionnel et entre `[...]` propose les ⟨**clés**⟩ suivantes (communes ou spécifiques) :

— ⟨**PosLabel**⟩ pour préciser la position (francisée) du label pour les points ; défaut : ⟨**b**⟩
— ⟨**StyleMarque**⟩ parmi ⟨**x/o**⟩ pour spécifier le style de la marque des points ; défaut : ⟨**o**⟩
— ⟨**TailleMarque**⟩ pour spécifier la taille de la marque des points (disque ou croix) ; défaut : ⟨**2pt**⟩
— ⟨**Couleur**⟩ pour paramétrer la couleur. défaut : ⟨**black**⟩

Les positions pour les labels des points sont *francisées* :

— ⟨**b**⟩ : `bas`
— ⟨**h**⟩ : `haut`
— ⟨**g**⟩ : `gauche`
— ⟨**d**⟩ : `droite`
— ⟨**hg**⟩ : `haut gauche`
— ...

De manière un peu plus spécifique :

— le `<label>` pour la commande `\PlacePointEspace` est optionnel, et est identique à `{nœud}` ;

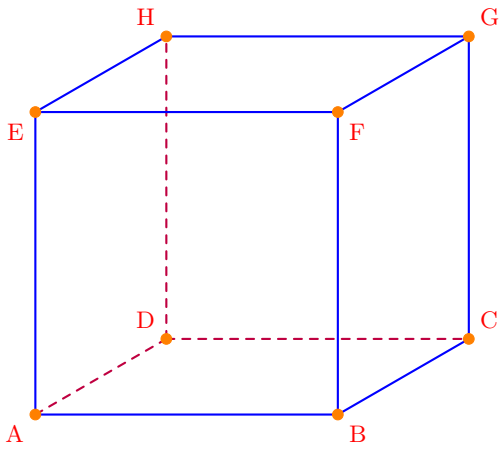
— la `{liste}` pour la commande `\PlacePointsEspace` est à donner – par exemple – sous la forme `A/0,0,0/bg B/5,2,1/hd` ;

— la `{liste}` pour la commande `\TraceSegmentsEspace` est à donner – par exemple – sous la forme `A/B A/C A/D B/D` ;

— les `[clés]` pour les segments correspondent aux options en langage Ti*k*Z.

Code LaTeX et sortie LaTeX

```
\begin{EnvTikzEspace}[VueClassique]
    %placement des points avec labels
    \PlacePointsEspace[Couleur=red]{A/0,0,0/bg B/4,0,0/bd C/4,4,0/hd D/0,4,0/hg
    ↪  E/0,0,4/bg F/4,0,4/bd G/4,4,4/hd H/0,4,4/hg}
    %segments pointillés
    \TraceSegmentsEspace[thick,dashed,purple]{A/D D/C D/H}
    %segments pleins
    \TraceSegmentsEspace[thick,blue]{A/B B/C C/G G/H H/E E/A E/F B/F F/G}
    %Marques points
    \MarquePointsEspace[Couleur=orange]{A,B,C,D,E,F,G,H}
\end{EnvTikzEspace}
```



4 Pavés

4.1 Commandes

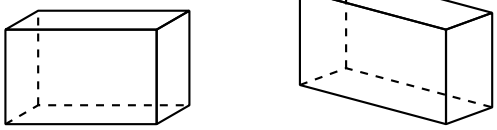
La commande *simplifiée* et *francisée* pour afficher un pavé (ou un cube !) est la commande `\PaveTikzTriDim`.

Code LaTeX

```
\begin{EnvTikzEspace}[Clés]<options tikz>
    \PaveTikzTriDim[clés]
\end{EnvTikzEspace}
```

Code LaTeX et sortie LaTeX

```
\begin{EnvTikzEspace}[VueClassique]
    \PaveTikzTriDim
\end{EnvTikzEspace}
\hspace{1cm}
\begin{EnvTikzEspace}
    \PaveTikzTriDim
\end{EnvTikzEspace}
```



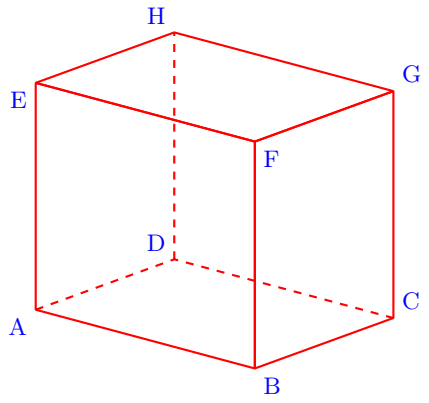
4.2 Clés et options

Quelques ⟨**clés**⟩ sont disponibles pour cette commande :

- ⟨**Largeur**⟩ : largeur du pavé ; défaut ⟨**2**⟩
- ⟨**Profondeur**⟩ : profondeur du pavé ; défaut ⟨**1**⟩
- ⟨**Hauteur**⟩ : hauteur du pavé ; défaut ⟨**1.25**⟩
- ⟨**Sommets**⟩ : liste des sommets (avec délimiteur § !) ; défaut ⟨**A§B§C§D§E§F§G§H**⟩
- ⟨**Math**⟩ : booléen pour forcer le mode math des sommets ; défaut ⟨**false**⟩
- ⟨**Epaisseur**⟩ : épaisseur des arêtes (en *langage simplifié* TikZ) ; défaut ⟨**thick**⟩
- ⟨**AffLabel**⟩ : booléen pour afficher les noms des sommets ; défaut ⟨**false**⟩
- ⟨**Plein**⟩ : booléen pour ne pas afficher les arêtes *invisibles* ; défaut ⟨**false**⟩
- ⟨**Cube**⟩ : booléen pour préciser qu'il s'agit d'un cube (seule ⟨**Largeur**⟩ est util(isé)e) ; défaut ⟨**false**⟩
- ⟨**Couleur**⟩ : couleur des arêtes ; défaut ⟨**black**⟩
- ⟨**CouleurSommets**⟩ : couleur des sommets. défaut ⟨**black**⟩

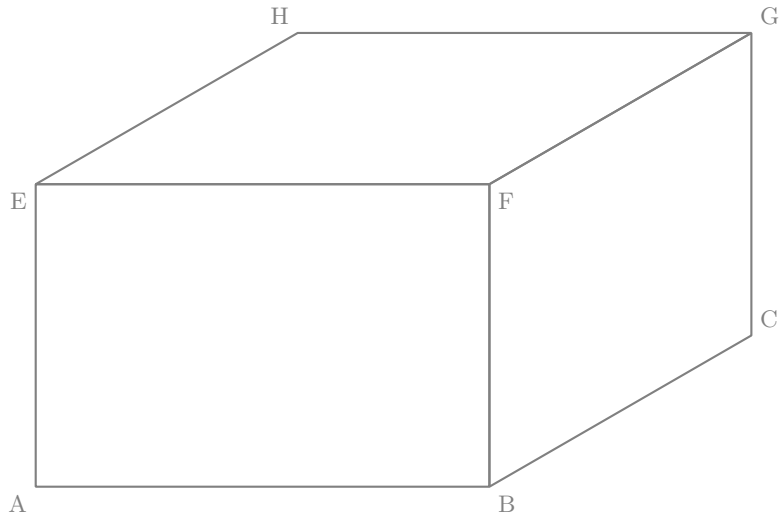
Code LaTeX et sortie LaTeX

```
\begin{EnvTikzEspace}
    \PaveTikzTriDim[Cube,Largeur=3,Couleur=red,CouleurSommets=blue,AffLabel]
\end{EnvTikzEspace}
```



Code LaTeX et sortie LaTeX

```
\begin{EnvTikzEspace}[VueClassique]<scale=2>
    \PaveTikzTriDim[Largeur=3,Profondeur=4,Hauteur=2,Plein,Couleur=gray,AffLabel,
    ↪  CouleurSommets=gray]
\end{EnvTikzEspace}
```



Quatrième partie

Empilements de cubes

5 Environnement dédié

L'environnement dédié à la création de figures type *empilement de cubes* est... `EmpilementCubes`.
Il existe deux manières de définir les empilements :

— en travaillant par *plaques* verticales de l'arrière vers l'avant ;

— en travaillant par les *hauteurs* des colonnes, de l'arrière vers l'avant **et** de gauche à droite.

L'unité de base des cubes est fixée au départ à 0,5 cm.

Les axes (et de ce fait la vue proposée !) sont fixés, non modifiables, donc cette partie est beaucoup moins performante que ce propose le package ProfCollege avec sa commande `\VueCubes` !

Code LaTeX

```
\begin{EmpilementCubes}[échelle]<options tikz>
    %commandes
\end{EmpilementCubes}
```

6 Création par *plaques*

6.1 Commandes

La commande pour créer une plaque *verticale* est `\PlaquePetitsCubes`, avec la contrainte de créer la *figure* de l'arrière vers l'avant.
Il existe également la commande `\PlaqueVide` pour *passer* une ligne.

Code LaTeX

```
\begin{EmpilementCubes}[échelle]<options tikz>
    \PlaquePetitsCubes[couleur(s)]{empilement}[épaisseur traits]
    \PlaqueVide[nb]
\end{EmpilementCubes}
```

Les plaques créées sont affichées l'une *devant* l'autre, et elles sont – par défaut – collées les unes aux autres.

6.2 Options et arguments

Le premier argument, optionnel et entre `[...]` permet de spécifier une couleur (`cyan` par défaut) ou une liste de couleurs qui seront utilisées pour la création des plaques :

- soit une couleur unique, qui sera *codée* par `1` pour la création des cubes ;
- soit plusieurs couleurs, sous la forme `couleur1/couleur2/couleur3/...` qui seront codées par `1`, `2`, ... pour la création des cubes.

Le second argument, obligatoire et entre `{...}` est quant à lui la liste, des lignes à construire, avec comme ordres :

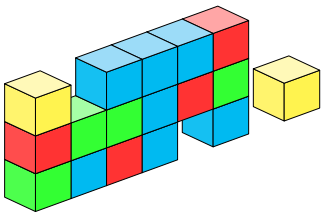
- du bas vers le haut (caractère de séparation `/`) ;
- de la gauche vers la droite (caractère de séparation `,`) ;
- un `-` code un *trou*, et un numéro code une couleur (comme définie(s) précédemment).

Le dernier argument, optionnel et entre `[...]` permet de spécifier une dimension pour les tracés (`0.2pt` par défaut).

Cette manière permet de créer des plaques avec couleurs *individuelles* et des *trous* éventuels.

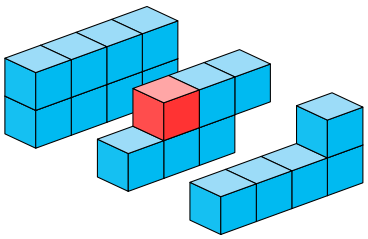
Code LaTeX et sortie LaTeX

```
\begin{EmpilementCubes}
    %plaque
    %de bas en haut : couleurs VBRB-B-J / RVVBRV / J-BBBR
    \PlaquePetitsCubes[cyan/red/green/yellow]{ 3121-1-4 / 233123 / 4-1112 }
\end{EmpilementCubes}
```



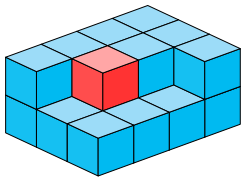
Code LaTeX et sortie LaTeX

```
\begin{EmpilementCubes}
    %plaque n°1 (fond)
    \PlaquePetitsCubes{ 1111 / 1111 }[0.4pt]
    \PlaqueVide[2]
    %plaque n°2
    \PlaquePetitsCubes[cyan/red]{ 111- / -211  }[0.4pt]
    \PlaqueVide[2]
    %plaque n°3 (devant)
    \PlaquePetitsCubes{ 1111 / ---1 }[0.4pt]
\end{EmpilementCubes}
```



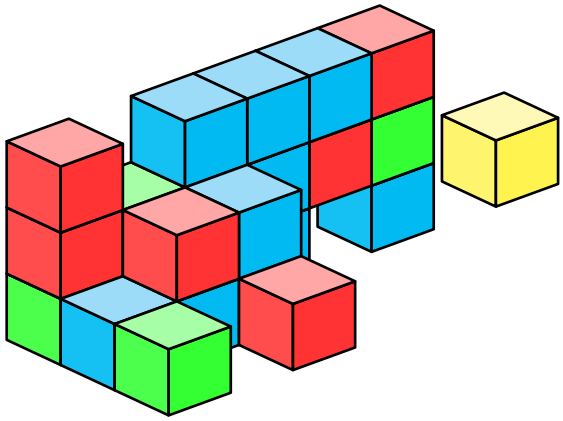
Code LaTeX et sortie LaTeX

```
\begin{EmpilementCubes}
    %plaque n°1 (fond)
    \PlaquePetitsCubes{ 1111 / 1111 }
    %plaque n°2
    \PlaquePetitsCubes[cyan/red]{ 111- / -211  }
    %plaque n°3 (devant)
    \PlaquePetitsCubes{ 1111 / ---1 }
\end{EmpilementCubes}
```



Code LaTeX et sortie LaTeX

```
\begin{EmpilementCubes}[1.75]
    \PlaquePetitsCubes[cyan/red/green/yellow]{ 3111-1-4 / 233123 / 2-1112 }[1pt]
    \PlaquePetitsCubes[cyan/red]{ 111 / -21 }[1pt]
    \PlaquePetitsCubes[cyan/red/green]{ 3-2 }[1pt]
\end{EmpilementCubes}
```



7 Création par *hauteurs*

7.1 Commande

L'idée, reprise du package ProfCollege[3] permet d'afficher une empilement de cubes (monochromes, et sans trou) en précisant – grâce à un système de *grille* – les hauteurs des colonnes. La commande qui permet de réaliser cet empilement est `\BlocPetitsCubes`.

Code LaTeX

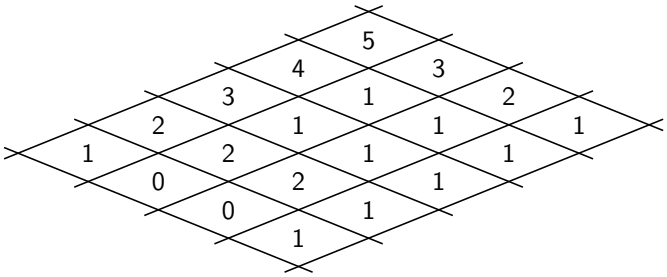
```
%création dans un environnement dédié
\begin{EmpilementCubes}[échelle]<options tikz>
    \BlocPetitsCubes*[couleur]{grille des hauteurs}[epaisseur traits]
\end{EmpilementCubes}
```

Code LaTeX

```
%création autonome
\BlocPetitsCubes[couleur]{grille des hauteurs}[epaisseur traits]
```

Le support de la grille des hauteurs est donc à donner sous forme *rectangulaire*, en respectant un nombre *homogène* de colonnes par ligne !

Voici une grille permettant d'anticiper la création d'un assemblage (en spécifiant les hauteurs) :



7.2 Options et arguments

Le premier argument, optionnel et entre `[...]` permet de spécifier une couleur (`cyan` par défaut).
Le second argument, obligatoire et entre `{...}` est quant à lui la liste des hauteurs, comme présentée précédemment :

- le caractère de séparation des *plaques* est le `/` ;
- pour chaque plaque, le caractère de séparation des colonnes est le `,`.

Le dernier argument, optionnel et entre `[...]` permet de spécifier une dimension pour les tracés (`0.2pt` par défaut).

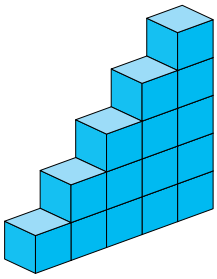
Par exemple, la *grille* associée à l'empilement précédent est :

`1,2,3,4,5 / 0,2,1,1,3 / 0,2,1,1,2 / 1,1,1,1,1`

3. https://www.ctan.org/pkg/profcollege

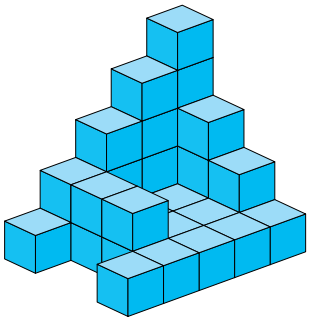
Code LaTeX et sortie LaTeX

```
\begin{EmpilementCubes}
    \BlocPetitsCubes*{1,2,3,4,5}
\end{EmpilementCubes}
```



Code LaTeX et sortie LaTeX

```
%commande autonome, taille par défaut
\BlocPetitsCubes{1,2,3,4,5 / 0,2,1,1,3 / 0,2,1,1,2 / 1,1,1,1,1}
```



Code LaTeX et sortie LaTeX

```
\begin{EmpilementCubes}[2]
    \BlocPetitsCubes*[violet]{1,2,3,4,5 / 0,2,1,1,3 / 0,2,1,1,2 / 1,1,1,1,1}[0.8pt]
\end{EmpilementCubes}
```