PixelArtTikz [fr]

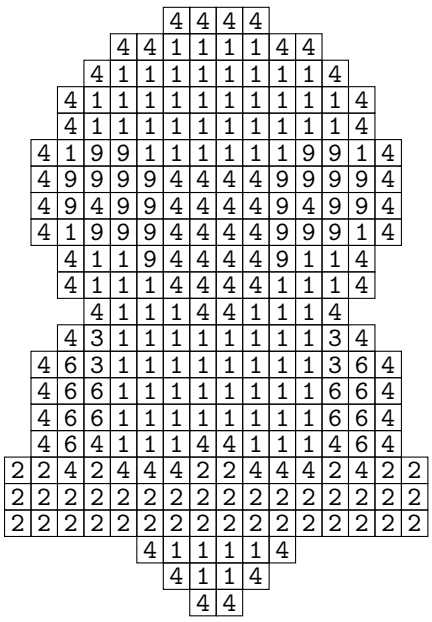
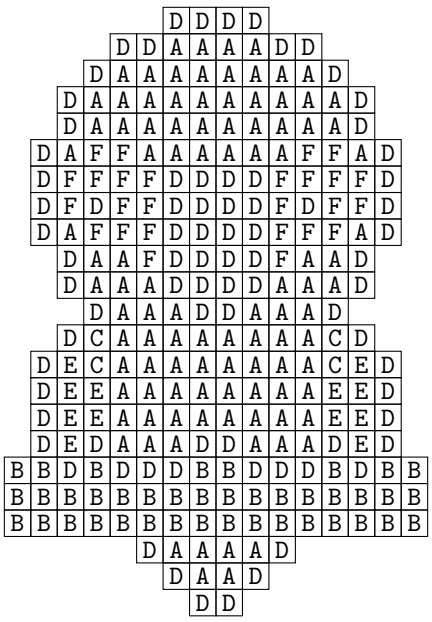
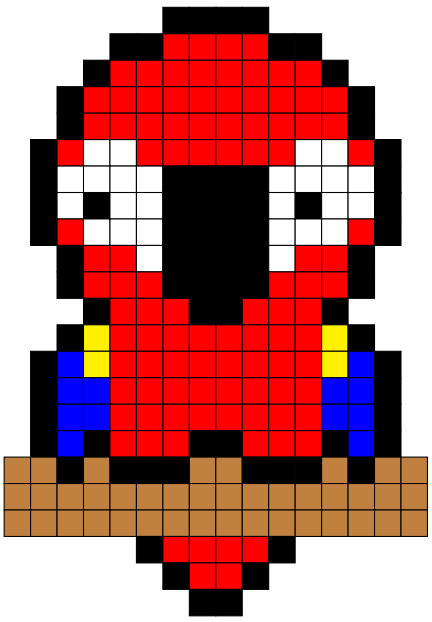
Des PixelArts, en Ti*k*Z,

avec solution et couleurs.

Version 0.1.2 - 12 octobre 2023

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

- Des commandes pour afficher des PixelArts.
- Environnement pour compléter éventuellement le PixelArt.

LaTeX

pdfLaTeX

LuaLaTeX

TikZ

TeXLive

MiKTeX

Table des matières

Première partie

Introduction

1 Le package PixelArtTikz

1.1 Introduction

L'idée est de *proposer*, dans un environnement Ti*k*Z, une commande permettant de générer des grilles PixelArt.
Les données sont *lues* à partir d'un fichier csv, externe au fichier tex ou déclaré en interne grâce à l'environnement `filecontents`.

Avant toute chose, quelques petites infos sur les données au format csv, surtout dans l'optique de sa lecture et de son traitement par les commandes :

— le fichier de données csv doit être formaté avec le séparateur décimal « , » ;

— des cases vides seront codées par « - ».

Le fichier csv peut être déclaré directement dans le fichier tex, grâce à l'environnement filecontents (intégré en natif sur les dernières versions de LaTeX) :

Code LaTeX

```
\begin{filecontents*}{nomfichier.csv}
    A,B,C,D
    A,B,D,C
    B,A,C,D
    B,A,D,C
\end{filecontents*}
```

À la compilation, le fichier nomfichier.csv sera créé automatiquement, et l'option **⟨[overwrite]⟩** permet (logiquement) de propager les modifications au fichier csv.

1.2 Chargement du package, et option

Le package *central* est ici csvsimple, qui permet de lire et traiter le fichier csv.
Il est « disponible » en version LaTeX 2_ε ou en version LaTeX3. Par défaut, PixelArtTikz le charge en version LaTeX3, mais une option est disponible pour une *rétro-compatibilité* avec la version LaTeX 2_ε.

L'option **⟨[csvii]⟩** permet de passer l'appel au package en version LaTeX 2_ε.

Code LaTeX

```
\usepackage{PixelArtTikz}                    %chargement du package version 3
%qui charge :
%\RequirePackage{expl3}
%\RequirePackage[l3]{csvsimple}

\usepackage[csvii]{PixelArtTikz}             %chargement du package version 2
%qui charge :
%\RequirePackage[legacy]{csvsimple}
```

1.3 Packages utilisés

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

Il charge les packages et librairies suivantes :

— `tikz`, `xintexpr` et `xinttools` ;

— `xstring`, `xparse`, `simplekv` et `listofitems`.

1.4 Commandes et environnement

Il existe deux manières de représenter un PixelArt :

— soit par une commande autonome et indépendante ;

— soit par un environnement Ti*k*Z dans lequel du code pourra être *rajouté*.

Code LaTeX

```
%Commande autonome
\PixelArtTikz[clés]<options tikz>{fichier.csv}

%Commande semi-autonome, à intégrer dans un environnement tikz
\PixelArtTikz*[clés]{fichier.csv}

%environnement
\begin{EnvPixelArtTikz}[clés]<options tikz>{fichier.csv}
    %code tikz
\end{EnvPixelArtTikz}
```

2 Les couleurs

Concernant les couleurs, l'utilisateur utilisera celles disponibles avec les packages chargés.

Les couleurs disponibles sans autre package sont donc :

magenta	cyan	blue	green	red	darkgray	olive	lime	brown	lightgray
white	gray	black	yellow	violet	teal	purple	pink	orange	

3 Petit aparté sur les fichiers csv

CSV désigne un format de fichiers dont le rôle est de présenter des données séparées par des virgules. Il s'agit d'une manière simplifiée d'afficher des données afin de les rendre transmissibles d'un programme à un autre.

Dans notre cas, le fichier csv contiendra les *codes* qui seront analysés un par un et ligne par ligne pour avoir le rendu par *code*, *symbole* ou *couleur*.

Il doit être préparé avec des caractères (codes) *simples* pour que le code de PixelArtTikz puisse fonctionner.

Deuxième partie

Commandes et environnement

4 La commande principale

4.1 Exemple introductif

La commande `\PixelArtTikz` nécessite de connaître :

— le fichier csv à traiter ;

— la liste (en fait sous forme de chaîne) des codes utilisés dans le fichier csv (comme `234679` ou `ABCDJK...`) ;

— la liste des symboles (éventuellement !) à afficher dans les cases s'il y a ambiguïté, comme `25,44,12` ou `AA,AB,AC` ;

— la liste des couleurs (si la correction est demandée), dans le même ordre que la liste des caractères.

On peut donc commencer par créer le fichier csv qui sera lu et interprété par les commandes du package. Le fichier peut-être créé directement dans la code du fichier tex.

Code LaTeX

```
%déclaration du fichier csv
\begin{filecontents*}[overwrite]{base.csv}
    A,B,C,D
    A,B,D,C
    B,A,D,C
    C,A,B,D
\end{filecontents*}
```

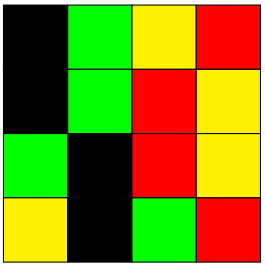
Code LaTeX

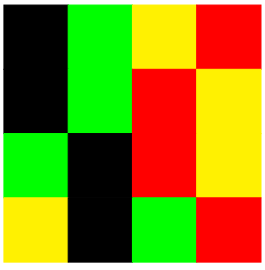
```
%notice et PixelArt
\begin{center}
    \begin{tblr}{colspec={*{4}{Q[1.25cm,c,m]}},hlines,vlines,rows={1.15em}}
        \SetCell[c=4]{c} Notice & & & \\
        A & B & C & D \\
        45 & 22 & 1 & 7 \\
        Noir & Vert & Jaune & Rouge \\
    \end{tblr}
\end{center}

\PixelArtTikz[Codes=ABCD,Style=\large\sffamily,Unite=0.85]{base.csv}
~~
\PixelArtTikz[Codes=ABCD,Symboles={45,22,1,7},Symb,Style=\large\sffamily,Unite=0.85]{base.csv}
~~
\PixelArtTikz[Codes=ABCD,Couleurs={black,green,yellow,red},Correction,Unite=0.85]{base.csv}
~~
\PixelArtTikz[Codes=ABCD,Couleurs={black,green,yellow,red},Correction,BordCases=false,Unite=0.85]{base.csv}
```

Notice			
A	B	C	D
45	22	1	7
Noir	Vert	Jaune	Rouge

A	B	C	D
A	B	D	C
B	A	D	C
C	A	B	D

45	22	1	7
45	22	7	1
22	45	7	1
1	45	22	7





4.2 Clés et options

Code LaTeX

```
\PixelArtTikz[clés]<options tikz>{fichier.csv}
```

Le premier argument, *optionnel* et entre `[...]` propose des Clés nécessaires au bon fonctionnement de la commande :

- la clé ⟨**Codes**⟩ contient la *chaîne* des codes *simples* du fichier csv ;
- la clé ⟨**Couleurs**⟩ qui contient la *liste* des couleurs associées ;
- la clé ⟨**Symboles**⟩ qui contient la *liste éventuelles* des caractères alternatifs à afficher dans les cases ;
- la clé booléenne ⟨**Correction**⟩ qui permet de colorier le PixelArt ; défaut false
- la clé booléenne ⟨**Symb**⟩ qui permet d'afficher les caractères *alternatifs* ; défaut false
- la clé booléenne ⟨**BordCases**⟩ qui permet d'afficher les bords des cases de la correction ; défaut true
- la clé ⟨**Style**⟩ qui permet de spécifier le style des caractères. défaut scriptsize

Le second argument, *optionnel* et entre `<...>` sont des options – en langage Ti*k*Z – à passer à l'environnement qui sert de base au PixelArt.

Le troisième argument, *obligatoire*, est le nom du fichier csv à utiliser.

On rappelle que le fichier peut être créé au préalable, et placé dans le répertoire du fichier, ou bien il peut être créé *en direct*, à l'aide du package filecontents (chargé par LaTeX).

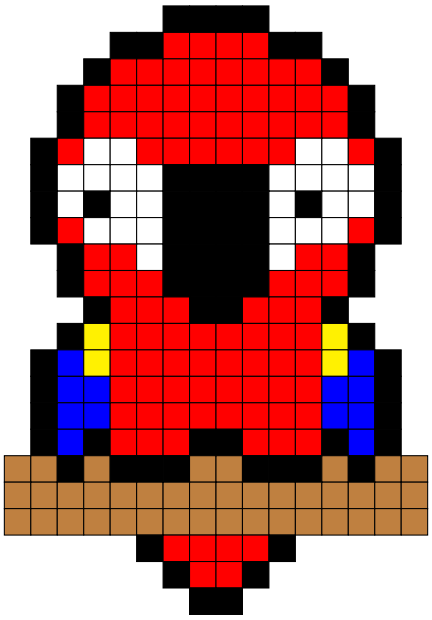
Code LaTeX

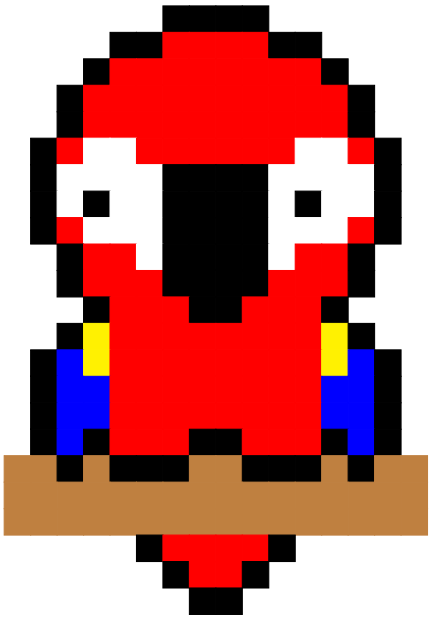
```
%création du fichier csv
\begin{filecontents*}[overwrite]{test1.csv}
    -,-,-,-,-,-,4,4,4,4,-,-,-,-,-,-
    -,-,-,-,4,4,1,1,1,1,4,4,-,-,-,-
    -,-,-,4,1,1,1,1,1,1,1,1,4,-,-,-
    -,-,4,1,1,1,1,1,1,1,1,1,1,4,-,-
    -,-,4,1,1,1,1,1,1,1,1,1,1,4,-,-
    -,4,1,9,9,1,1,1,1,1,1,9,9,1,4,-
    -,4,9,9,9,9,4,4,4,4,9,9,9,9,4,-
    -,4,9,4,9,9,4,4,4,4,9,4,9,9,4,-
    -,4,1,9,9,9,4,4,4,4,9,9,9,1,4,-
    -,-,4,1,1,9,4,4,4,4,9,1,1,4,-,-
    -,-,4,1,1,1,4,4,4,4,1,1,1,4,-,-
    -,-,-,4,1,1,1,4,4,1,1,1,4,-,-,-
    -,-,4,3,1,1,1,1,1,1,1,1,3,4,-,-
    -,4,6,3,1,1,1,1,1,1,1,1,3,6,4,-
    -,4,6,6,1,1,1,1,1,1,1,1,6,6,4,-
    -,4,6,6,1,1,1,1,1,1,1,1,6,6,4,-
    -,4,6,4,1,1,1,4,4,1,1,1,4,6,4,-
    2,2,4,2,4,4,4,2,2,4,4,4,2,4,2,2
    2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2
    2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2
    -,-,-,-,-,4,1,1,1,1,4,-,-,-,-,-
    -,-,-,-,-,-,4,1,1,4,-,-,-,-,-,-
    -,-,-,-,-,-,-,4,4,-,-,-,-,-,-,-
\end{filecontents*}
```

Code LaTeX

```
%codes simples et sans ambiguïté
%une case vide sera codée par -
\PixelArtTikz[Codes=123469,Style=\ttfamily,Unite=0.35]{test1.csv}
~~
\PixelArtTikz[Codes=123469,Couleurs={red,brown,yellow,black,blue,white},Correction,Unite=0.35]{test1.csv}
~~
\PixelArtTikz[Codes=123469,Couleurs={red,brown,yellow,black,blue,white},Correction,Unite=0.35,BordCases=false]%
    {test1.csv}
```

				4	4	4	4							
			4	4	1	1	1	1	4	4				
		4	1	1	1	1	1	1	1	1	4			
	4	1	1	1	1	1	1	1	1	1	1	4		
	4	1	1	1	1	1	1	1	1	1	1	4		
4	1	9	9	1	1	1	1	1	1	9	9	1	4	
4	9	9	9	9	4	4	4	4	9	9	9	9	4	
4	9	4	9	9	4	4	4	4	9	4	9	9	4	
4	1	9	9	9	4	4	4	4	9	9	9	1	4	
	4	1	1	9	4	4	4	4	9	1	1	4		
	4	1	1	1	4	4	4	4	1	1	1	4		
		4	1	1	1	4	4	1	1	1	4			
	4	3	1	1	1	1	1	1	1	1	3	4		
4	6	3	1	1	1	1	1	1	1	1	3	6	4	
4	6	6	1	1	1	1	1	1	1	1	6	6	4	
4	6	6	1	1	1	1	1	1	1	1	6	6	4	
4	6	4	1	1	1	4	4	1	1	1	4	6	4	
2	2	4	2	4	4	4	2	2	4	4	4	2	4	2	2
2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
					4	1	1	1	1	4				
						4	1	1	4					
							4	4						

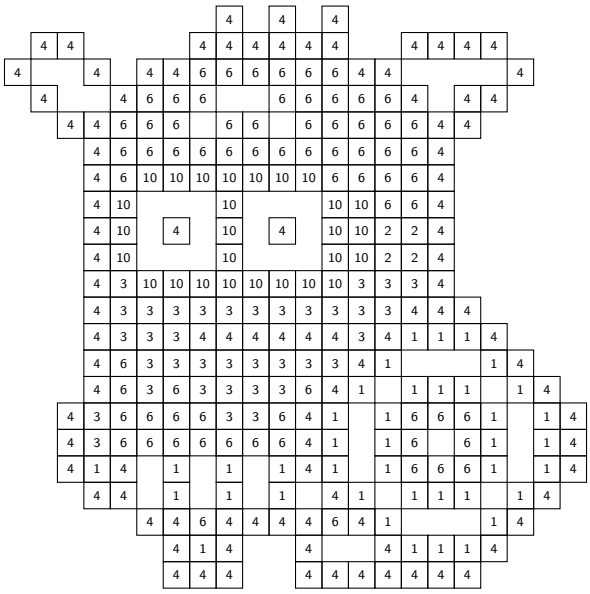


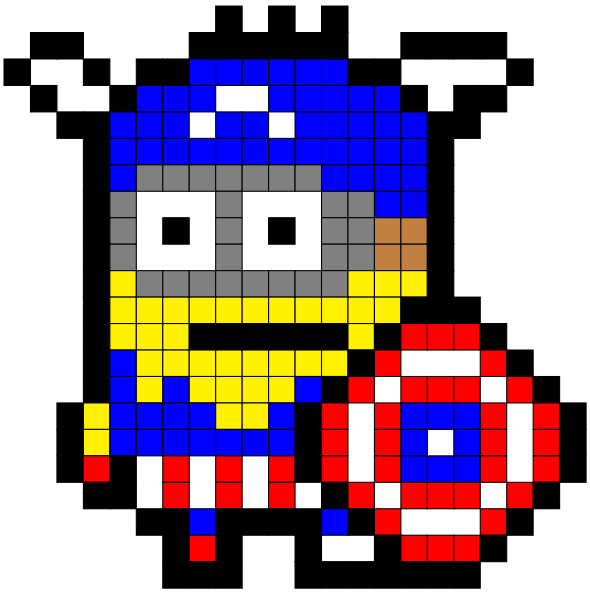


Dans l'exemple suivant, les *symboles* à afficher ne peuvent pas servir de *codes*, donc on utilise les options liées à ⟨**Symboles**⟩ pour s'affranchir de cette limitation.

Code LaTeX

```
%codes à afficher, avec utiliser des symboles

\begin{filecontents*}[overwrite]{cap.csv}
    -,-,-,-,-,-,-,-,D,-,D,-,D,-,-,-,-,-,-,-,-,-,-,-,-
    -,D,D,-,-,-,-,D,D,D,D,D,D,-,-,D,D,D,D,-,-,-
    D,-,-,D,-,D,D,F,F,F,F,F,F,D,D,-,-,-,-,D,-,-
    -,D,-,-,D,F,F,F,-,-,F,F,F,F,F,D,-,D,D,-,-,-
    -,-,D,D,F,F,F,-,F,F,-,F,F,F,F,F,D,D,D,-,-,-
    -,-,-,D,F,F,F,F,F,F,F,F,F,F,F,F,F,D,-,-,-,-
    -,-,-,D,F,J,J,J,J,J,J,F,F,F,F,D,-,-,-,-,-
    -,-,-,D,J,-,-,-,J,-,-,-,J,J,F,F,D,-,-,-,-
    -,-,-,D,J,-,D,-,J,-,D,-,J,J,B,B,D,-,-,-,-
    -,-,-,D,J,-,-,-,J,-,-,-,J,J,B,B,D,-,-,-,-
    -,-,-,D,C,J,J,J,J,J,J,J,C,C,C,D,-,-,-,-,-
    -,-,-,D,C,C,C,C,C,C,C,C,C,C,D,D,D,-,-,-,-
    -,-,-,D,C,C,C,D,D,D,D,D,C,D,A,A,A,D,-,-,-
    -,-,-,D,F,C,C,C,C,C,C,C,D,A,-,-,-,A,D,-,-
    -,-,-,D,F,C,F,C,C,C,C,F,D,A,-,A,A,A,-,A,D,-
    -,-,D,C,F,F,F,F,C,C,F,D,A,-,A,F,F,F,A,-,A,D
    -,-,D,C,F,F,F,F,F,F,F,D,A,-,A,F,-,F,A,-,A,D
    -,-,D,A,D,-,A,-,A,-,A,D,A,-,A,F,F,F,A,-,A,D
    -,-,-,D,D,-,A,-,A,-,A,-,D,A,-,A,A,A,-,A,D,-
    -,-,-,-,-,-,D,D,F,D,D,D,D,F,D,A,-,-,-,A,D,-,-
    -,-,-,-,-,-,-,D,A,D,-,-,D,-,-,D,A,A,A,D,-,-,-
    -,-,-,-,-,-,-,D,D,D,-,-,D,D,D,D,D,D,D,-,-,-,-
\end{filecontents*}

\PixelArtTikz[Codes=ABCDFJ,Symboles={1,2,3,4,6,10},Symb,Style=\tiny\sffamily,Unite=0.35]{cap.csv}
~~
\PixelArtTikz[Codes=ABCDFJ,Couleurs={red,brown,yellow,black,blue,gray},Correction,Unite=0.35]{cap.csv}
```





4.3 Commande étoilée

La commande *étoilée* `\PixelArtTikz*` permet d'intégrer le PixelArt dans un environnement créé par l'utilisateur. Cela permet par exemple de pouvoir rajouter du code en parallèle du PixelArt.

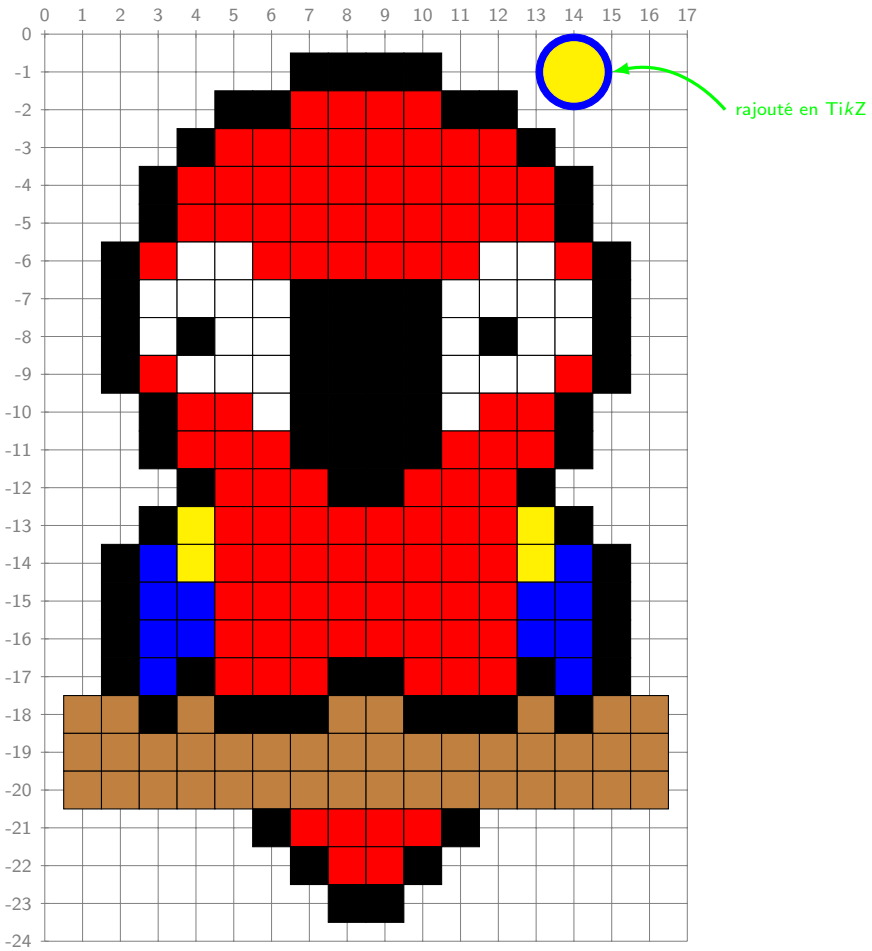
Il est à noter que, dans ce cas :

— l'argument *optionnel* entre `<...>` est inutile ;

— la clé ⟨**Unite**⟩ n'intervient plus dans le tracé (elle peut être passée directement dans l'environnement !)

Code LaTeX

```
\begin{center}
    \begin{tikzpicture}[scale=0.5]
        %grille pour visualiser le positionnement
        \draw[very thin,gray,xstep=1,ystep=1] (0,0) grid (17,-24) ;
        \foreach \x in {0,1,...,17} \draw[very thin,gray] (\x,-3pt)--(\x,3pt)%
        node[above,font=\scriptsize\sffamily] {\x} ;
        \foreach \y in {0,-1,...,-24} \draw[very thin,gray] (3pt,\y)--(-3pt,\y)%
        node[left,font=\scriptsize\sffamily] {\y} ;
        %le PixelArt
        \PixelArtTikz*[Codes=123469,Couleurs={red,brown,yellow,black,blue,white},Correction]{test1.csv}
        %du code rajouté
        \filldraw[blue] (14,-1) circle[radius=1] ;
        \filldraw[yellow] (14,-1) circle[radius=0.8] ;
        \draw[green,very thick,<-,>=latex] (15,-1) to[bend left=30] (18,-2)%
        node[right,font=\scriptsize\sffamily] {rajouté en Ti\textit{k}Z} ;
    \end{tikzpicture}
\end{center}
```



5 Environnement PixelArt

5.1 Commande et options

Le package PixelArtTikz propose également un environnement pour créer un PixelArt, et pouvoir rajouter des éléments en marge du PixelArt.

— L'environnement est créé autour de Ti*k*Z et le code rajouté le sera dans un langage Ti*k*Z !

— Le code rajouté le sera, dans ce cas, *au-dessus* du PixelArt !

Le fonctionnement global est le même que pour la commande autonome.

Code LaTeX

```
\begin{EnvPixelArtTikz}[clés]<options tikz>{fichier.csv}
    %code(s) tikz qui seront au-dessus du PixelArt
\end{EnvPixelArtTikz}
```

Le premier argument, *optionnel* et entre `[...]` propose des Clés nécessaires au bon fonctionnement de la commande :

— la clé **⟨Codes⟩** contient la *chaîne* des codes *simples* du fichier csv ;

— la clé **⟨Couleurs⟩** qui contient la *liste* des couleurs associées ;

— la clé **⟨Symboles⟩** qui contient la *liste éventuelles* des caractères alternatifs à afficher dans les cases ;

— la clé booléenne **⟨Correction⟩** qui permet de colorier le PixelArt ; défaut false

— la clé booléenne **⟨Symb⟩** qui permet d'afficher les caractères *alternatifs* ; défaut false

— la clé booléenne **⟨BordCases⟩** qui permet d'afficher les bords des cases de la correction ; défaut true

— la clé **⟨Style⟩** qui permet de spécifier le style des caractères. défaut scriptsize

Le second argument, *optionnel* et entre `<...>` sont des options – en langage Ti*k*Z – à passer à l'environnement qui sert de base au PixelArt.

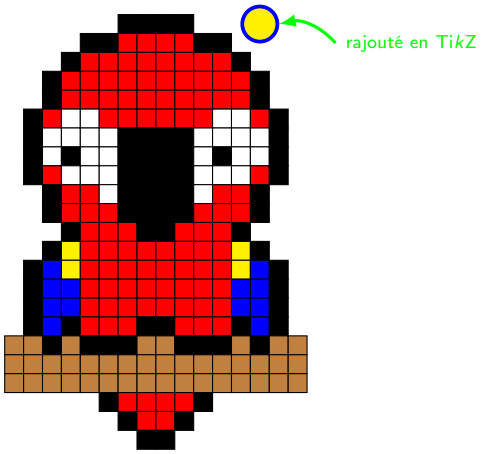
Le troisième argument, *obligatoire*, est le nom du fichier csv à utiliser.

5.2 Exemple

Les symboles affichés dans les cases sont situés aux nœuds de coordonnées $(c\,;\,-l)$ où l et c sont les numéros de ligne et de colonne correspondants à la position de la donnée dans le fichier csv.

Code LaTeX

```
\begin{center}
    \begin{EnvPixelArtTikz}%
        [Codes=123469,Couleurs={red,brown,yellow,black,blue,white},Correction,Unite=0.25]
        {test1.csv}
        \filldraw[blue] (14,-1) circle[radius=1] ;
        \filldraw[yellow] (14,-1) circle[radius=0.8] ;
        \draw[green,very thick,<-,>=latex] (15,-1) to[bend left=30] (18,-2)%
        node[right,font=\scriptsize\sffamily] {rajouté en Ti\textit{k}Z} ;
    \end{EnvPixelArtTikz}
\end{center}
```



6 La commande pour un *mini*-PixelArt

6.1 Idée

L'idée est de proposer une commande pour insérer, sans passer par un fichier csv, un petit PixelArt avec une liste de couleurs réduite.

Code LaTeX
```
\MiniPixelArt[clés]{liste des couleurs}
```

Le premier argument, *optionnel* et entre `[...]` propose des Clés nécessaires au bon fonctionnement de la commande :

- la clé ⟨**Unite**⟩ pour spécifier l'unité des cases ; défaut 0.25em
- le booléen ⟨**Bord**⟩ pour afficher une bordure aux cases. défaut false

Le deuxième argument, obligatoire et entre `{...}` permet de donner les couleurs des cases :

- chaque couleur est codée par une lettre :
  - R : rouge
  - B : bleu
  - N : noir
  - . : blanc
  - O : orange
  - V : vert
  - J : jaune
  - G : gris
  - M : marron
  - P : violet
- chaque *passage à la ligne* est spécifié par `,` ;
- les bords éventuels ont une épaisseur égale à 10 % de l'unité des carreaux.

Le dernier argument, optionnel et entre `<...>`, permet quant à lui de passer des options à l'environnement tikz créé.

6.2 Exemples

Code LaTeX
```
\MiniPixelArt{%
    ..RR..RR..,
    .RRRRRRRR.,
    RRRRRRRRRR,
    RRRRRRRRRR,
    RRRRRRRRRR,
    .RRRRRRRR.,
    ..RRRRRR..,
    ...RRRR...,
    ....RR....
}
```

Code LaTeX
```
En ligne, on a \MiniPixelArt[Unite=5mm,Bord]{NBVOJV,JGP.NR}<baseline=(current bounding box.center)> ce miniPA.
```

En ligne, on a  ce miniPA.

Troisième partie

Historique

`v0.1.2` : Possibilité de créer des *mini*-PixelArts
`v0.1.1` : Correction d'un bug avec les couleurs
`v0.1.0` : Version initiale