

PixelArtTikz [fr]

Des PixelArts, en TikZ,
avec solution et couleurs.

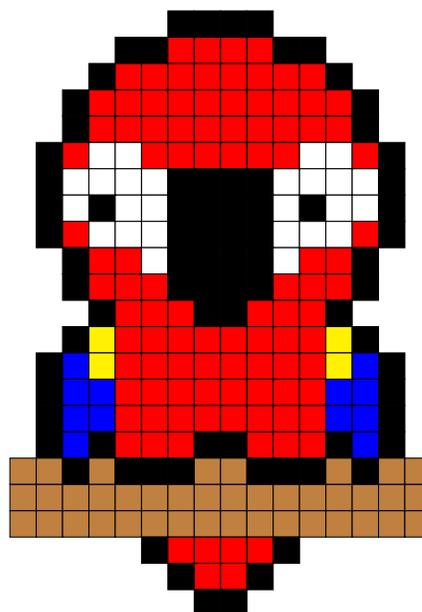
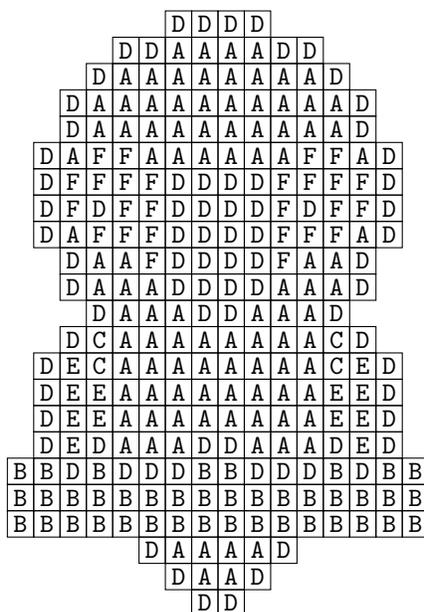
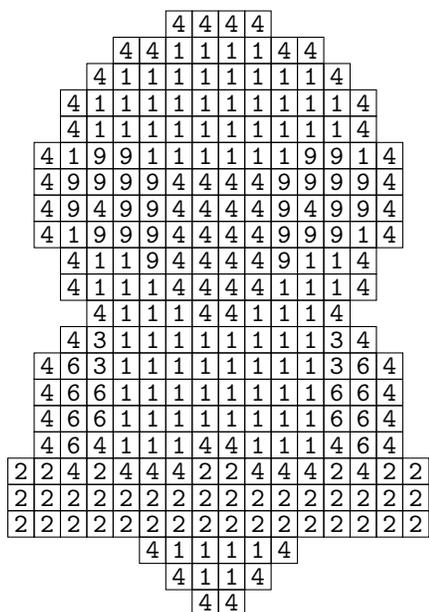
Version 0.1.8 - 5 mai 2025

Cédric Pierquet

c.pierquet - at - outlook . fr

<https://forge.apps.education.fr/pierquetcedric/packages-latex>

- Des commandes pour afficher des PixelArts.
- Des commandes pour découper des PixelArts en plusieurs parties.
- Environnement pour compléter éventuellement le PixelArt.
- Des PixelArts avec anamorphose cylindrique, des *mini*-PixelArts.



L^AT_EX

pdfL^AT_EX

LuaL^AT_EX

TikZ

T_EXLive

MiK_TE_X

Table des matières

I	Introduction	3
1	Le package PixelArtTikz	3
1.1	Introduction	3
1.2	Chargement du package, et option	3
1.3	Packages utilisés	3
1.4	Commandes et environnement	4
2	Compléments	4
2.1	Les couleurs	4
2.2	Petit aparté sur les fichiers csv	4
II	Commandes principales	5
3	La commande principale	5
3.1	Exemple introductif	5
3.2	Clés et options	6
3.3	Symboles dans une liste	9
3.4	Commande étoilée	10
4	Environnement PixelArt	11
4.1	Commande et options	11
4.2	Exemple	11
5	La commande pour découper	12
5.1	Idée et fonctionnement global	12
5.2	Aide quant à la création du découpage	12
5.3	Affichage d'une partie unique	14
5.4	Création automatique du découpage	15
III	Commandes complémentaires	17
6	PixelArt et anamorphose cylindrique	17
6.1	Idée	17
6.2	Clés et options	18
6.3	Exemple avec données inversées (Yoda)	19
6.4	Exemple avec données classiques (Sorcière)	21
7	La commande pour un <i>mini</i>-PixelArt	22
7.1	Idée	22
7.2	Exemples	22
8	Création automatique du tableau notice	23
8.1	Idée	23
8.2	Clés et exemples	23
8.3	Commande simplifiée (??) pour les cases	24
9	Avec le package datatool	25
9.1	Commandes	25
9.2	Exemple	25
IV	Historique	26

Première partie

Introduction

1 Le package PixelArtTikz

1.1 Introduction

L'idée est de *proposer*, dans un environnement `TikZ`, 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) :

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

Code \LaTeX

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 $\text{\LaTeX} 2_{\epsilon}$ ou en version $\text{\LaTeX} 3$. Par défaut, `PixelArtTikz` le charge en version $\text{\LaTeX} 3$, mais une option est disponible pour une *rétro-compatibilité* avec la version $\text{\LaTeX} 2_{\epsilon}$.

L'option `([csvii])` permet de passer l'appel au package en version $\text{\LaTeX} 2_{\epsilon}$.

```
\usepackage{PixelArtTikz}           %chargement du package version 3
%qui charge :
%\RequirePackage{expl3}
%\RequirePackage[13]{csvsimple}

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

Code \LaTeX

À noter qu'une version alternative *simple*, avec le package `datatool`, est également disponible.

1.3 Packages utilisés

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

Il charge les packages et bibliothèques suivantes :

- `tikz`, `xintexpr` et `xinttools` ;
- `xstring`, `simplekv` et `listofitems` ;
- `multicol` (pour le *découpage*).

1.4 Commandes et environnement

Il existe trois manières de représenter un PixelArt :

- soit par une commande autonome et indépendante ;
- soit par un environnement TikZ dans lequel du code pourra être *rajouté* ;
- soit par *découpage* de la grille en plusieurs (travail collaboratif).

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}
```

Code \LaTeX

```
%Affichage d'un bloc précis (si découpage)
\PixelArtTikzBloc[clés]<options tikz>{fichier.csv}{découpage}{num bloc}

%Affichage des blocs (si découpage)
\DecoupPixelArtTikz(*)[clés]<options tikz>{fichier.csv}{découpage}

%Affichage d'une 'aide'
\AideGrillePixelArtTikz(*)[Echelle]{fichier.csv}{découpage}
```

2 Compléments

2.1 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	

Pour des couleurs *francisées*, le package `couleurs-fr` pourra être utile.

2.2 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 principales

3 La commande principale

3.1 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

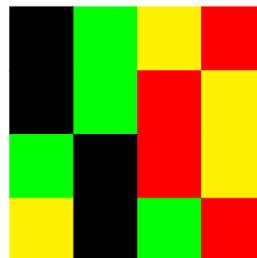
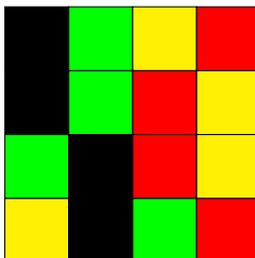
```
%notice et PixelArt
\begin{center}
\begin{tblr}{colspec={*4}{Q[1.25cm,c,m]},hlines,vlines,rows={1.15em}}
\SetCell[c=4]{c} Notice & & & \\\
A & B & C & D \\\
45 & 22 & 1 & 7 \\\
Noir & Vert & Jaune & Rouge \\\
\end{tblr}
\end{center}

\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



3.2 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 **<Decoupage>** pour afficher des lignes de découpage éventuel :
 - sous la forme **<<nb lig bloc>x<nb col bloc>>** pour spécifier la dimension des blocs ;
 - sous la forme **<<nb blocs V>+<nb blocs H>>** pour spécifier le nombre de blocs ;
- la clé **<Style>** qui permet de spécifier le style des caractères. défaut scriptsize

Le second argument, *optionnel* et entre <...> sont des options – en langage TikZ – 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

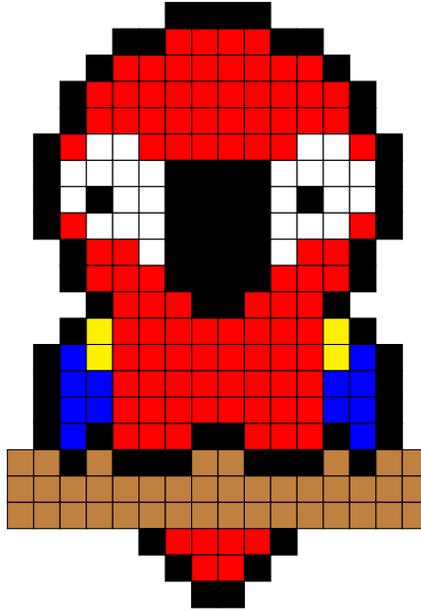
```
%création du fichier csv
\begin{filecontents*}[overwrite]{test1.csv}
-,,-,-,-,4,4,4,4,-,-,-,-,-
-,,-,-,4,4,1,1,1,1,4,4,-,-,-,-
-,,-,4,1,1,1,1,1,1,1,1,4,-,-,-
-,,-,4,1,1,1,1,1,1,1,1,1,4,-,-
-,,-,4,1,1,1,1,1,1,1,1,1,1,4,-,-
-,,-,4,1,1,1,1,1,1,1,1,1,1,4,-,-
-,4,1,9,9,1,1,1,1,1,1,9,9,1,4,-
-,4,9,9,9,9,4,4,4,4,9,9,9,9,4,-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

```

%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	4						
	4	1	1	1	1	1	1	1	4						
	4	1	1	1	1	1	1	1	4						
4	1	9	9	1	1	1	1	1	9	9	1	4			
4	9	9	9	9	4	4	4	4	9	9	9	4			
4	9	4	9	9	4	4	4	4	9	4	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	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											



3.3 Symboles dans une liste

À noter qu'il est possible de donner comme symboles des listes dans lesquelles seront choisies aléatoirement les symboles.

Code \LaTeX

%codes à afficher, avec utiliser des symboles "aléatoires" dans une liste

```
\begin{filecontents*}[overwrite]{testlist.csv}
A,B,C,A
A,B,B,C
B,A,C,B
C,A,B,C
\end{filecontents*}

\textbf{Notice : }

Multiples de 5 : Rouge\\
Multiples de 3 : Vert\\
Multiples de 2 : Bleu

\PixelArtTikz[Codes=ABC,Symboles={5§25§35,3§9§21§27,2§4§8§14§16},Symb,Style=\large\sffamily,Unite=0.85]{testlist.csv}
\hspace{5mm}
\PixelArtTikz[Codes=ABC,Symboles={5§25§35,3§9§21§27,2§4§8§14§16},Symb,Style=\large\sffamily,Unite=0.85]{testlist.csv}
\hspace{5mm}
\PixelArtTikz[Codes=ABC,Couleurs={red,green,blue},Correction,Style=\large\sffamily,Unite=0.85]{testlist.csv}
```

Notice :

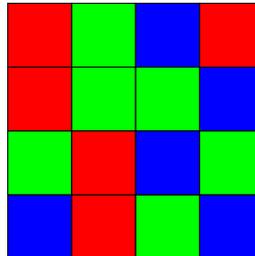
Multiples de 5 : Rouge

Multiples de 3 : Vert

Multiples de 2 : Bleu

5	3	4	25
25	3	9	4
3	5	14	27
8	35	21	8

25	3	2	25
5	9	3	4
3	5	2	3
4	25	27	2



3.4 Commande étoilée

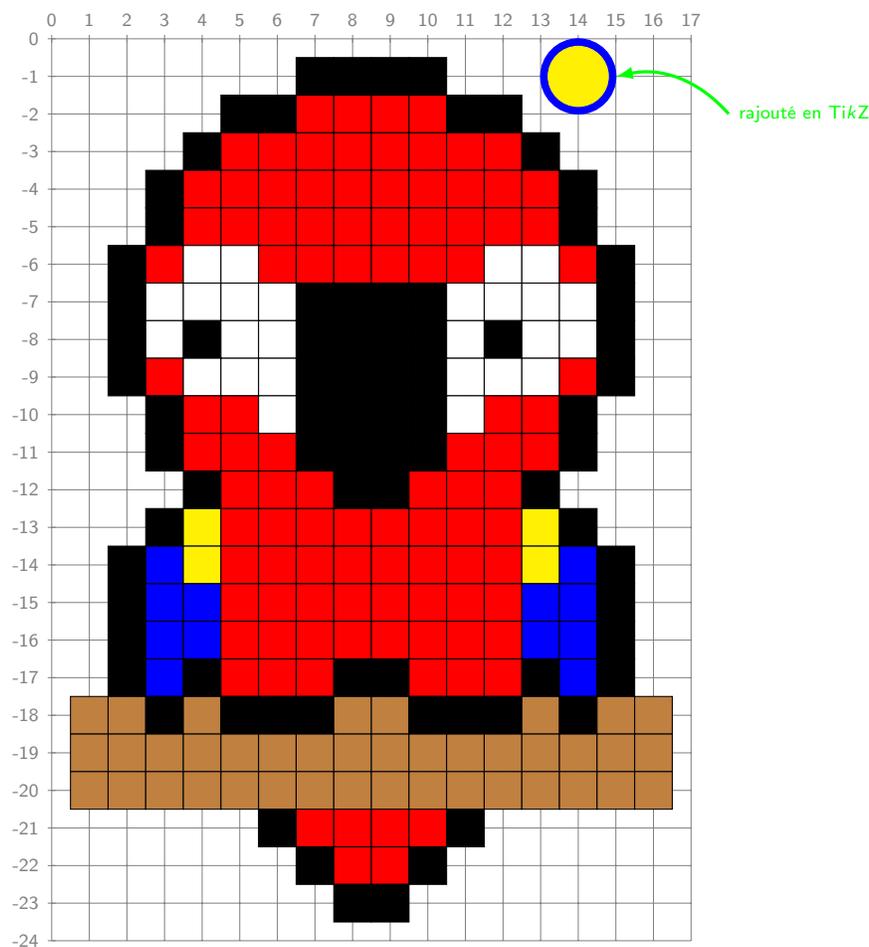
La commande étoilée `\PixelArtTikz*` permet d'intégrer le PixelArt dans un environnement créé par l'utilisateur. Cela permet par exemple de pouvoir rajouter du code en parallèle du PixelArt.

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}
```



4 Environnement PixelArt

4.1 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 `TikZ` et le code rajouté le sera dans un langage `TikZ`!
- Le code rajouté le sera, dans ce cas, *au-dessus* du `PixelArt`!

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

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

Code \LaTeX

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 `TikZ` – à passer à l'environnement qui sert de base au `PixelArt`.

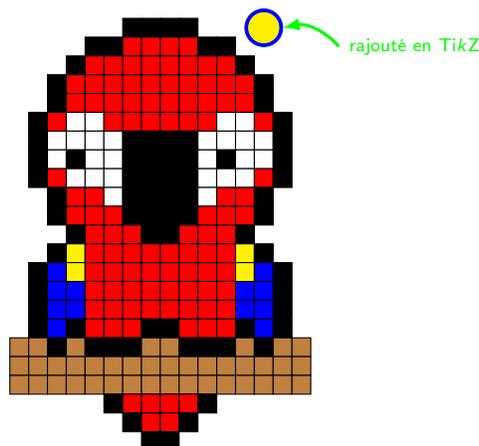
Le troisième argument, *obligatoire*, est le nom du fichier `csv` à utiliser.

4.2 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`.

```
\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}
```

Code \LaTeX



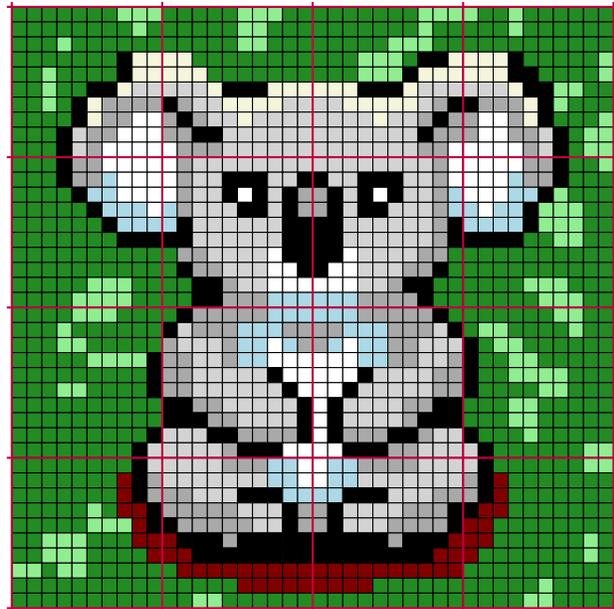

```
%découpage par blocs de taille 8x8
\AideGrillePixelArtTikz{PAkoala.csv}{8x8}
~~
\PixelArtTikz[Correction,Unite=0.125,Codes=ABCDEFGH,I,Couleurs={\listcoukkoala},Decoupage=8x8]{PAkoala.csv}
```

A1	A2	A3	A4	A5
B1	B2	B3	B4	B5
C1	C2	C3	C4	C5
D1	D2	D3	D4	D5
E1	E2	E3	E4	E5



```
%découpage par 4 blocs / ligne et 4 blocs / colonne
\AideGrillePixelArtTikz*[2]{PAkoala.csv}{4+4}
~~
\PixelArtTikz[Correction,Unite=0.2,Codes=ABCDEFGH,I,Couleurs={\listcoukkoala},Decoupage=4+4]{PAkoala.csv}
```

1.1	1.2	1.3	1.4
2.1	2.2	2.3	2.4
3.1	3.2	3.3	3.4
4.1	4.2	4.3	4.4



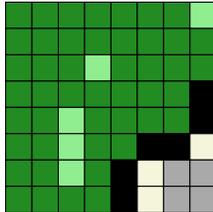
5.3 Affichage d'une partie unique

Une commande est disponible pour l'affichage d'une bloc particulier du PixelArt.
Les clés sont héritées de celle des commandes principales.

Code \LaTeX

```
%bloc 1/1 pour un découpage 8x8
\PixelArtTikzBloc[Unite=0.35,Codes=ABCDEFGH]{PAkoala.csv}{8x8}{1/1}
~~
\PixelArtTikzBloc[Unite=0.35,Codes=ABCDEFGH,Correction,Couleurs={\listcoulkoala}]{PAkoala.csv}{8x8}{1/1}
```

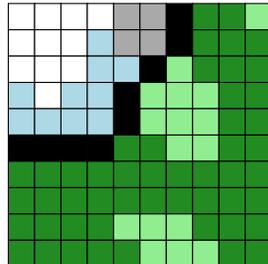
A	A	A	A	A	A	A	H
A	A	A	A	A	A	A	A
A	A	A	H	A	A	A	A
A	A	A	A	A	A	A	B
A	A	H	A	A	A	A	B
A	A	H	A	A	B	B	E
A	A	H	A	B	E	D	D
A	A	A	A	B	E	D	D



Code \LaTeX

```
%bloc 2/4 pour un découpage 4+4
\PixelArtTikzBloc[Unite=0.35,Codes=ABCDEFGH]{PAkoala.csv}{4+4}{2/4}
~~
\PixelArtTikzBloc[Unite=0.35,Codes=ABCDEFGH,Correction,Couleurs={\listcoulkoala}]{PAkoala.csv}{4+4}{2/4}
```

F	F	F	F	D	D	B	A	A	H
F	F	F	G	D	D	B	A	A	A
F	F	F	G	G	B	H	A	A	A
G	F	G	G	B	H	H	H	A	A
G	G	G	B	H	H	H	A	A	A
B	B	B	B	A	A	H	H	A	A
A	A	A	A	A	A	A	A	A	A
A	A	A	A	A	A	A	A	A	A
A	A	A	A	H	H	H	A	A	A
A	A	A	A	H	H	H	A	A	A



L'idée est ensuite d'utiliser cette commande d'insertion d'un bloc pour créer l'énoncé avec les grilles de découpage.
Mais il existe une commande de création *automatique* des grilles découpées !

5.4 Création automatique du découpage

Il existe également une commande dédiée pour le découpage automatique et la présentation automatique des petites grilles.

Code \LaTeX

%découpage par blocs de 8x8

\DecoupPixelArtTikz [Unite=0.3, Codes=ABCDEFGH] {PAkoala.csv} {8x8}

Grille A1

A	A	A	A	A	A	A	H
A	A	A	A	A	A	A	A
A	A	A	H	A	A	A	A
A	A	A	A	A	A	A	B
A	A	H	A	A	A	A	B
A	A	H	A	A	B	B	E
A	A	H	A	B	E	D	D
A	A	A	B	E	D	D	D

Grille A2

H	H	A	A	A	A	A	H
H	A	A	A	A	A	A	H
A	A	A	A	A	A	A	H
E	E	E	E	A	A	A	A
E	E	E	E	E	A	A	A
C	C	C	C	C	E	B	B
D	D	B	D	C	C	E	E
F	F	D	B	D	C	C	E

Grille A3

H	H	A	A	A	A	A	A
H	A	A	A	A	A	A	A
H	A	A	A	A	A	A	A
A	A	A	A	A	A	A	H
A	A	A	A	A	A	A	H
B	E	E	E	E	E	E	B
E	E	C	C	C	C	C	E
C	C	C	C	C	C	C	C

Grille A4

A	A	A	H	H	H	H	A
A	A	A	H	A	A	A	A
A	H	H	H	A	A	A	A
H	H	A	B	B	E	E	E
H	H	B	E	E	E	E	E
B	B	E	C	C	C	C	C
E	E	E	C	D	B	B	D
E	C	D	B	D	D	F	

Grille A5

A	H	A	A	A	A	A	A
A	H	A	A	A	A	A	A
A	A	A	A	A	A	A	A
E	B	A	A	A	A	A	H
E	B	B	A	A	A	A	H
C	E	B	B	H	H	A	A
D	D	E	B	H	A	A	A
F	D	E	B	A	A	A	H

Grille B1

A	A	A	B	C	D	D	F
A	A	A	B	C	D	F	F
A	A	A	B	D	D	F	F
A	A	A	B	D	D	G	F
A	A	A	A	B	G	G	F
A	A	A	A	A	B	G	G
A	A	A	A	A	A	B	G
A	A	A	A	A	A	A	B

Grille B2

F	F	D	D	B	B	C	C
F	F	F	D	D	D	C	C
F	F	F	D	C	C	C	C
F	F	F	D	C	C	B	B
F	F	F	D	C	C	B	F
G	F	G	D	C	C	C	B
G	G	B	B	C	C	C	C
B	B	B	B	C	C	C	C

Grille B3

C	C	C	C	C	C	C	C
C	C	C	C	C	C	C	C
C	C	C	C	C	C	C	C
B	C	C	B	B	C	C	B
B	C	B	D	D	B	C	B
B	C	B	D	D	B	C	B
C	C	B	B	B	B	C	C
C	C	B	B	B	B	C	C

Grille B4

C	C	C	B	D	D	D	F
C	C	C	D	D	F	F	F
C	C	C	C	D	F	F	F
B	B	C	C	D	F	F	F
F	B	C	C	D	G	F	F
B	C	C	C	D	G	G	F
C	C	C	C	D	B	G	G
C	C	C	C	D	B	B	B

Grille B5

F	D	D	C	B	A	H	H
F	F	D	C	B	A	H	H
F	F	D	D	B	A	A	H
F	G	D	D	B	A	A	A
F	G	G	B	H	A	A	A
G	G	B	H	H	H	A	A
G	G	B	H	H	H	A	A
B	B	A	A	H	H	A	A

Grille C1

A	A	A	A	A	A	A	A
A	A	A	A	A	A	A	A
A	A	A	A	H	H	H	H
A	A	A	A	H	H	H	H
A	A	A	A	H	H	H	A
A	A	A	H	A	A	A	A
A	A	A	H	A	A	A	A
A	A	A	A	A	H	H	H

Grille C2

A	A	A	D	D	C	C	C
A	A	A	B	D	D	C	C
A	A	A	A	B	B	D	C
A	A	A	A	B	B	D	D
A	A	A	B	D	D	D	D
A	A	B	C	D	D	D	D
A	A	D	C	C	D	G	
H	B	D	C	C	D	G	

Grille C3

C	F	B	B	B	B	F	C
C	F	F	B	B	F	F	C
C	F	F	F	F	F	F	C
D	G	G	G	G	G	G	D
D	G	G	G	G	G	G	D
G	G	D	D	D	D	G	G
G	F	G	D	D	G	F	G
G	F	F	F	F	F	F	G

Grille C4

C	C	C	D	D	A	A	A
C	C	D	D	B	A	A	A
C	D	B	B	A	A	A	A
D	D	D	B	A	A	A	A
D	D	D	D	B	A	A	A
G	D	D	D	C	B	A	H
G	D	D	C	C	B	A	H
G	D	D	C	C	D	B	A

Grille C5

A	A	A	A	A	A	A	A
A	A	A	A	A	A	A	A
A	A	H	H	H	A	A	A
A	A	A	H	H	H	A	A
A	A	A	H	H	H	A	A
H	A	A	A	H	A	A	A
H	H	A	A	A	A	A	A
A	H	H	A	A	A	A	A

Grille D1

A	A	A	A	A	A	A	A
A	A	A	H	H	A	A	A
A	A	A	A	A	A	A	A
A	A	A	A	A	A	A	A
A	A	A	A	A	A	A	A
A	A	A	A	A	A	A	A
A	A	A	A	A	A	A	A
A	A	A	A	A	A	A	I

Grille D2

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

Grille D3

C	C	B	F	F	F	F	B
C	C	B	F	F	F	B	C
C	C	C	B	F	B	C	C
D	D	D	B	F	B	D	D
B	B	F	F	F	F	B	B
D	G	G	F	F	G	G	D
C	G	G	F	F	G	G	C

Grille D4

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

Grille D5

A	H	H	H	H	A	A	A
A	A	A	H	H	A	A	A
A	H	H	A	A	A	A	A
A	H	H	A	A	A	A	A
A	A	A	A	A	A	H	H
A	A	A	A	A	A	H	H
I	A	A	A	A	A	H	H
I	A	A	A	A	A	A	A

Grille E1

A	A	A	A	A	A	A	I
A	A	A	A	A	H	A	I
A	A	A	A	H	H	H	H
A	A	H	H	H	A	A	A
A	A	H	H	H	A	A	A
H	H	A	H	H	A	A	A
A	A	A	A	A	A	A	A
A	A	A	A	A	A	A	A

Grille E2

B	D	D	D	C	C	C	B
I	B	D	D	D	D	D	C
I	B	B	D	D	D	D	D
I	I	I	B	B	B	D	B
A	I	I	I	B	B	B	B
A	A	A	I	I	I	I	I
A	A	A	A	A	A	A	I
A	A	A	A	H	H	A	A

Grille E3

B	B	G	G	G	G	B	B
C	C	B	D	D	B	C	C
C	C	B	D	D	B	C	C
B	B	B	B	B	B	B	B
B	B	B	B	B	B	B	B
I	I	I	I	I	I	I	I
I	I	I	I	I	I	I	I
A	A	A	A	A	A	A	A

Grille E4

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

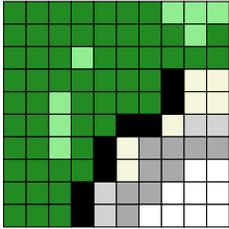
Grille E5

I	A	A	A	A	A	A	A
I	A	A	A	A	A	A	A
A	A	A	A	A	A	A	A
A	A	A	A	A	A	H	A
A	A	A	A	A	H	A	A
A	A	A	A	A	A	A	A
A	A	A	A	A	A	A	A
A	A	H	H	A	A	A	A

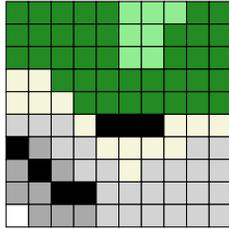
```
%découpage en4 blocs par 4
```

```
\DecoupPixelArtTikz[Unite=0.3,Codes=ABCDEFGH,I,Correction,Couleurs={\listcoulkoala}]{PAkoala.csv}{4+4}
```

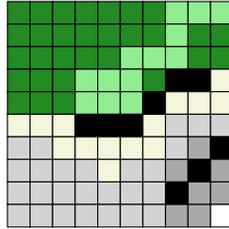
Grille A1



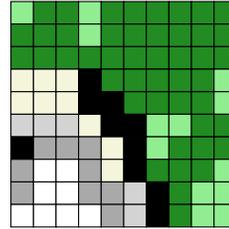
Grille A2



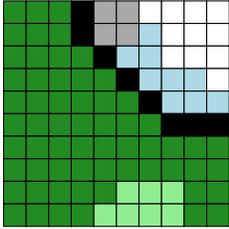
Grille A3



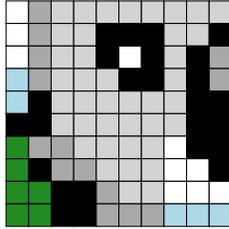
Grille A4



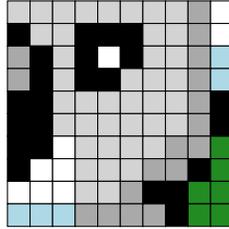
Grille B1



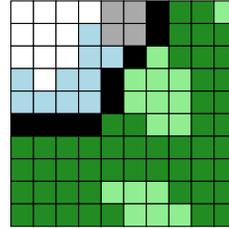
Grille B2



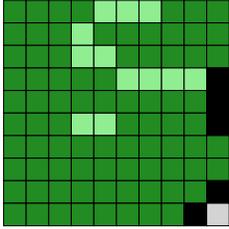
Grille B3



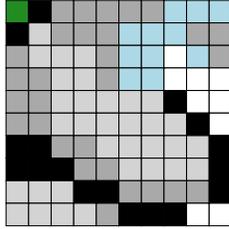
Grille B4



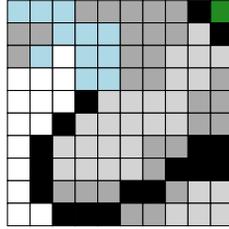
Grille C1



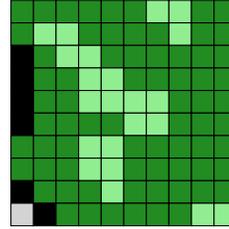
Grille C2



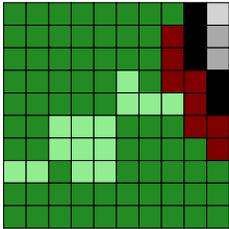
Grille C3



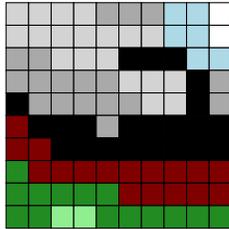
Grille C4



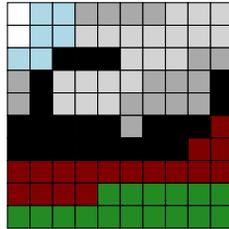
Grille D1



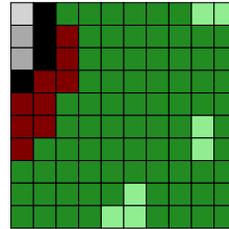
Grille D2



Grille D3



Grille D4



Troisième partie

Commandes complémentaires

6 PixelArt et anamorphose cylindrique

6.1 Idée

L'idée est de proposer de quoi créer un PixelArt dans le but d'utiliser une anamorphose cylindrique.

Sur <https://www.youtube.com/watch?v=PT8KUozBg3I>, il y a une vidéo *démonstration*, proposée par Jean-Yves Labouche.

Le fonctionnement global est similaire à celui de la commande *principale*, il existe cependant quelques ajustements :

- la possibilité de donner le fichier csv en mode *normal* ou *inversé* ;
- les dimensions (largeur & milieu) sont à préciser pour produire le PixelArt ;
- la commande est autonome (pour le moment) donc pas d'ajout(s) ultérieurement.

```
\PixelArtTikzCylindre[clés]{fichier.csv}
```

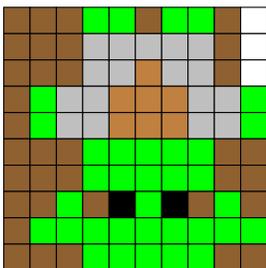
Code \LaTeX

Les fichiers illustrant ce paragraphe sont donnés ci-dessous.

```
%version avec données inversées
\begin{filecontents*}[overwrite]{PAYoda.csv}
E,E,E,A,A,E,A,A,E,D
E,E,E,F,F,F,F,F,E,D
E,E,E,F,F,C,C,F,F,E,D
E,A,F,F,C,C,C,F,F,A
E,A,F,F,C,C,C,F,F,A
E,E,E,A,A,A,A,A,E,E
E,E,E,A,A,A,A,A,E,E
E,E,A,E,B,A,B,E,A,E
E,A,A,A,A,A,A,A,A,A
E,E,E,A,A,A,A,A,E,E
\end{filecontents*}

\PixelArtTikz[%
  Codes=ABCDEF,
  Couleurs={green,black,brown,white,brown!75!black,lightgray},
  Correction,Unite=0.35]%
{PAYoda.csv}
```

Code \LaTeX

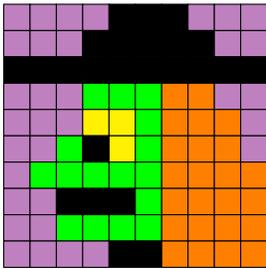


```

%version avec données normales
\begin{filecontents*}{PASorciere.csv}
V,V,V,V,N,N,N,V,V,V
V,V,V,N,N,N,N,N,V,V
N,N,N,N,N,N,N,N,N,N
V,V,V,G,G,O,O,V,V
V,V,V,J,J,G,O,O,O,V
V,V,G,N,J,G,O,O,O,V
V,G,G,G,G,G,O,O,O,O
V,V,N,N,N,G,O,O,O,O
V,V,G,G,G,G,O,O,O,O
V,V,V,V,N,N,O,O,O,O
\end{filecontents*}

\PixelArtTikz[%
  Codes=VNGOJ,
  Couleurs={violet!50,black,green,orange,yellow},
  Correction,Unite=0.35]%
{PASorciere.csv}

```



6.2 Clés et options

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

- la clé **<Largeur>** qui définit la largeur (rayon en cm) du rendu ; défaut 6
- la clé **<Centre>** qui définit la largeur (rayon en cm) du *milieu* ; défaut 1.25
- 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 **<Style>** qui permet de spécifier le style des caractères. défaut `normalsize`
- 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 **<Solution>** qui permet d'afficher la solution (avec effet *miroir*) ; défaut `false`
- la clé booléenne **<Swap>** qui permet de spécifier le type de données (**<true>** := normal ; **<false>** := inversé). défaut `false`

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

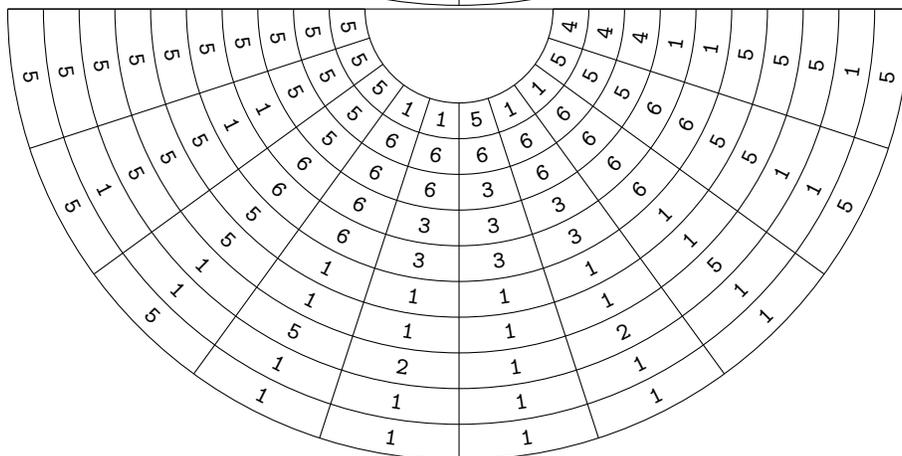
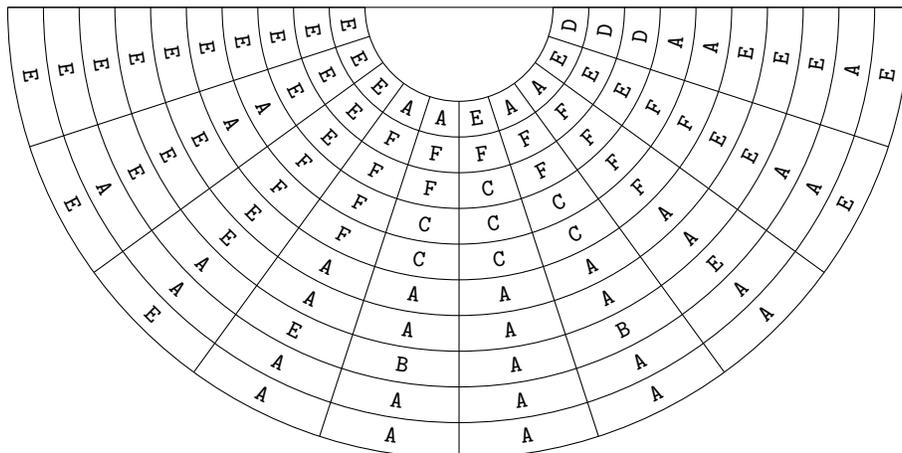
6.3 Exemple avec données inversées (Yoda)

Dans ce paragraphe, on utilise les données PAYoda, qui correspondent à la disposition *inversée*, donc la clé **(Swap)** n'est pas nécessaire.

Code \LaTeX

```
%version classique
\PixelArtTikzCylindre[Codes=ABCDEF,Style=\small\ttfamily]{PAYoda.csv}

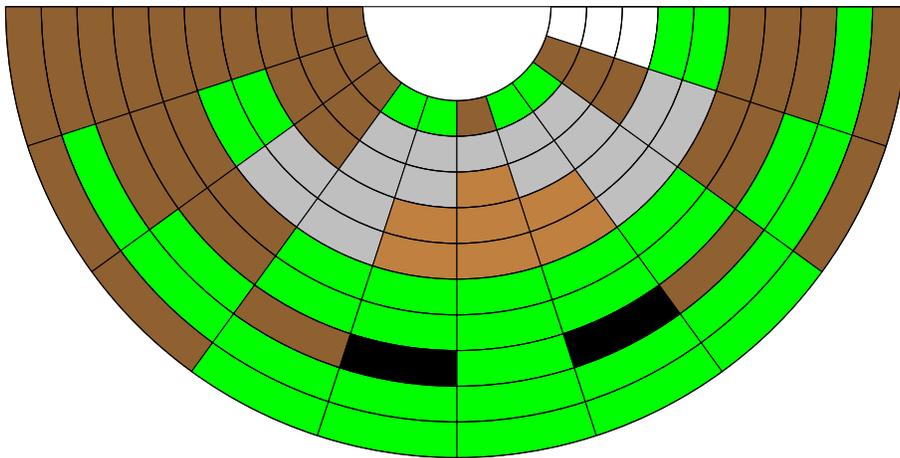
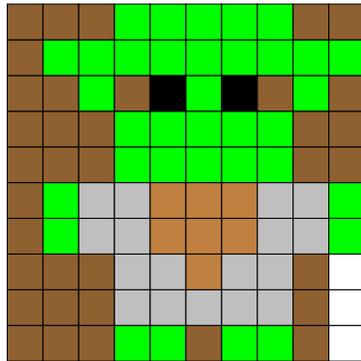
%version avec 'symboles'
\PixelArtTikzCylindre[Codes=ABCDEF,Symboles={1,2,3,4,5,6},Symb,Style=\small\ttfamily]{PAYoda.csv}
```



```

%Correction et solution
\begin{tabular}{c}
\PixelArtTikzCylindre[%
  Codes=ABCDEF,
  Couleurs={green,black,brown,white,brown!75!black,lightgray},
  Solution]%
  {PAYoda.csv}
\\
\PixelArtTikzCylindre[%
  Codes=ABCDEF,
  Couleurs={green,black,brown,white,brown!75!black,lightgray},
  Correction]%
  {PAYoda.csv}
\end{tabular}

```

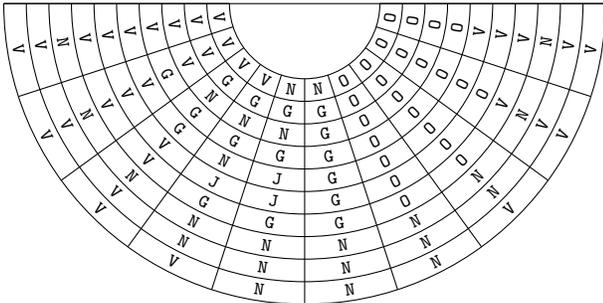


6.4 Exemple avec données classiques (Sorcière)

Dans ce paragraphe, on utilise les données `PAsorciere`, qui correspondent à la disposition *normale*, donc la clé `(Swap)` est nécessaire.

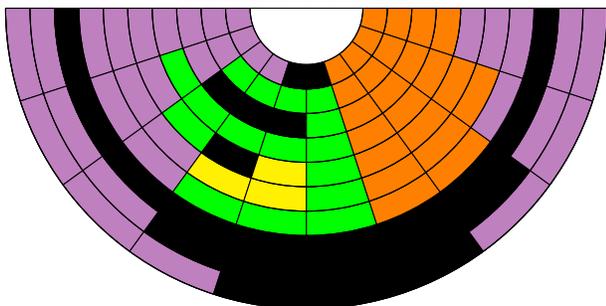
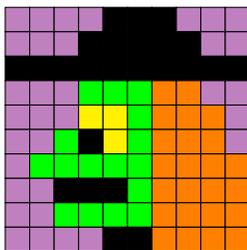
Code \LaTeX

```
%version classique
\PixelArtTikzCylindre[%
  Largeur=4,Centre=1,Codes=VNGOJ,
  Couleurs={violet!50,black,green,orange,yellow},
  Swap,Style=\ttfamily\scriptsize}%
{PAsorciere.csv}
```



Code \LaTeX

```
%Correction et solution
\begin{tabular}{c}
\PixelArtTikzCylindre[%
  Largeur=4,Centre=0.75,Codes=VNGOJ,
  Couleurs={violet!50,black,green,orange,yellow},
  Swap,Solution]%
{PAsorciere.csv}
\\
\PixelArtTikzCylindre[%
  Largeur=4,Centre=0.75,Codes=VNGOJ,
  Couleurs={violet!50,black,green,orange,yellow},
  Swap,Correction]%
{PAsorciere.csv}
\end{tabular}
```



7 La commande pour un *mini-PixelArt*

7.1 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.

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

Code \LaTeX

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.

7.2 Exemples

```
\MiniPixelArt{%  
..RR..RR..,  
.RRRRRRR.,  
RRRRRRRRR,  
RRRRRRRRR,  
RRRRRRRRR,  
.RRRRRRR.,  
..RRRRR.,  
...RRR.,  
...RR.,  
}  
  

```

Code \LaTeX

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

Code \LaTeX

En ligne, on a  ce miniPA.

8 Création automatique du tableau notice

8.1 Idée

L'idée est de proposer une commande pour créer automatiquement le tableau de notice, avec coloration des cases.

Le package utilisé est `tabulararray`, et le code propose deux présentations du tableau, sous forme horizontal ou vertical (les tableaux sur plusieurs lignes ne sont pas gérés...)

```
\TablCouleursPixelArt(*)[clés]{%
données1,%
données2,%
...
}
```

Code \TeX

Les `<données>` sont à mettre sous la forme `<NomCouleur>/<CodeCouleur>/<CouleurPolice>/<Résultat>`.

8.2 Clés et exemples

La version *étoilée* force le tableau en mode vertical.

Les `<clés>` disponibles pour cette commande sont :

- la clé `<Largeur>`, pour spécifier les informations de largeur :
 - sous la forme `<auto>` pour les adapter aux contenus (valeur par défaut) ;
 - sous la forme `<largeurglobale>` en mode H (les cases auront la même largeur) ;
 - sous la forme `<largeurcolonne>` ou `<largeurcolonne1/largeurcolonne2>` en mode V ;
- la clé `<Police>` spécifier une police particulière.

```
%par défaut
\TablCouleursPixelArt{%
Marron/BrunIntense/Blanc/75,%
{Gris Clair}/GrisClair/Noir/{112,5},%
Vert Clair/VertClair/Noir/600,%
Noir/Noir/Blanc/9,%
Beige/Beige/Noir/15,%
Gris foncé/GrisFonce/Noir/{202,5},%
Bleu clair/BleuClair/Noir/288,%
Vert foncé/VertFonce/Blanc/10,%
Blanc/Blanc/Noir/55%
}
```

Code \TeX

Marron	Gris Clair	Vert Clair	Noir	Beige	Gris foncé	Bleu clair	Vert foncé	Blanc
75	112,5	600	9	15	202,5	288	10	55

```
%personnalisations
\TablCouleursPixelArt[Largeur=\linewidth,Police=\small\sffamily]{%
Marron/BrunIntense/Blanc/75,%
{Gris Clair}/GrisClair/Noir/{112,5},%
Vert Clair/VertClair/Noir/600,%
Noir/Noir/Blanc/9,%
Beige/Beige/Noir/15,%
Gris foncé/GrisFonce/Noir/{202,5},%
Bleu clair/BleuClair/Noir/288,%
Vert foncé/VertFonce/Blanc/10,%
Blanc/Blanc/Noir/55%
}
```

Code \TeX

Marron	Gris Clair	Vert Clair	Noir	Beige	Gris foncé	Bleu clair	Vert foncé	Blanc
75	112,5	600	9	15	202,5	288	10	55

```
%personnalisations
\TablCouleursPixelArt*[Largeur=3cm/]{%
  Marron/BrunIntense/Blanc/75,%
  {Gris Clair}/GrisClair/Noir/{112,5},%
  Vert Clair/VertClair/Noir/600,%
  Noir/Noir/Blanc/9,%
  Beige/Beige/Noir/15,%
  Gris foncé/GrisFonce/Noir/{202,5},%
  Bleu clair/BleuClair/Noir/288,%
  Vert foncé/VertFonce/Blanc/10,%
  Blanc/Blanc/Noir/55%
}
```

Marron	75
Gris Clair	112,5
Vert Clair	600
Noir	9
Beige	15
Gris foncé	202,5
Bleu clair	288
Vert foncé	10
Blanc	55

8.3 Commande simplifiée (??) pour les cases

Il est également possible de créer le tableau *manuellement*, avec une commande *simplifiée* pour la création des cases.

```
%dans un environnement tblr, chargé avec [expand=\expanded] et \expanded !
\cctblr[couleur police]{couleur fond}{case}
```

```
\begin{tblr}[expand=\expanded]{width=\linewidth, colspec={*{5}{Q[m,1]}}, hlines, vlines}
  \expanded{\cctblr[Blanc]{BrunIntense}{Marron}} &
  \expanded{\cctblr[GrisClair]{Gris clair}} &
  \expanded{\cctblr[VertClair]{Vert clair}} &
  \expanded{\cctblr[Blanc]{Noir}{Noir}} &
  \ldots \\
  75 & {112,5} & 600 & 9 & \ldots \\
\end{tblr}
```

Marron	Gris clair	Vert clair	Noir	...
75	112,5	600	9	...

9 Avec le package datatool

9.1 Commandes

Le package `datatool`, chargé par `PixelArtTikz`, permet de représenter un PixelArt (sans environnement). Les clés sont les mêmes que pour la version `csvsimple`, mais il est nécessaire de *traiter* le fichier `csv` en amont des tracés.

Code `MTX`

```
%lecture du fichier csv
\readdtcsv{fichier.csv}{nomlecture}

%traitement du PixelArt complet
\dtpixlarttikz[clés]{nomlecture}

%traitement du PixelArt en mode découpage
\dtpixlarttikzblock[clés]{nomlecture}{LxC ou L+C}{numbloc}
```

À noter que la commande est autonome, ne permet pas d'ajouts éventuels, mais est compatible avec le traitement par découpage (mais sans création automatique des blocs de découpage avec présentation).

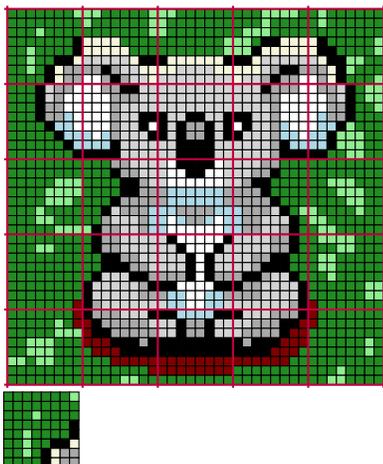
9.2 Exemple

Code `MTX`

```
%lecture du fichier csv (koala)
\readdtcsv{PAkoala.csv}{DTkoala}

%traitement du PixelArt complet
\dtpixlarttikz[Correction,Unite=0.125,Codes=ABCDEFGH,I,Couleurs={\listcoulkoala},Decoupage=8x8]{DTkoala}

%traitement du PixelArt en mode découpage
\dtpixlarttikzblock[Unite=0.125,Correction,Codes=ABCDEFGH,I,Couleurs={\listcoulkoala}]%
{DTkoala}{8x8}{1/1}
```



Quatrième partie

Historique

- v0.1.8 : Bugfix
- v0.1.7 : Correction d'un bug avec les découpages
- v0.1.6 : Ajout d'un style pour les traits + utilisation de `datatool` (plus rapide?)
- v0.1.5 : Symboles sous forme de liste(s) (éléments tirés aléatoirement) + Amélioration du traitement
- v0.1.4 : PixelArts avec anamorphose cylindrique
- v0.1.3 : Possibilité de créer des PixelArts collaboratifs
- v0.1.2 : Possibilité de créer des *mini*-PixelArts
- v0.1.1 : Correction d'un bug avec les couleurs
- v0.1.0 : Version initiale