L'extension pour LaTeX

scratch

v 0.3

8 aout 2017

Christian Tellechea
unbonpetit@netc.fr

Cette extension permet de dessiner des empilements de blocs similaires à ceux que l'on trouve dans le logiciel de programmation *visuelle* « scratch[1] ».

1. Le logiciel que l'on peut utiliser en ligne à https://scratch.mit.edu/

1 Avant propos

La présente extension scratch requiert les extensions simplekv et tikz. Celles-ci sont automatiquement chargées par scratch.
Fidèle à mes convictions, la documentation de cette extension n'est disponible qu'en français.

2 L'environnement scratch

Pour dessiner un programme comme le fait scratch, il faut ouvrir un environnement « scratch » et écrire dans cet environnement les macros correspondant aux *blocs* que l'on veut y mettre :

```
\begin{scratch}
    macros pour dessiner des blocs
\end{scratch}
```

Comme le savent ceux qui enseignent l'algorithmique et la programmation avec le très-à-la-mode [2] logiciel « scratch », les programmes sont construits avec des briques, appelés « blocs », qui peuvent s'emboiter les uns sur les autres. Ces blocs sont de plusieurs couleurs, chacune correspondant à un type d'instruction que l'on retrouve dans les menus de scratch.
J'ai pris le parti d'écrire des macros ayant comme argument le texte qui figure dans le bloc. Ce faisant, on a plus de liberté que dans scratch où les blocs ont des textes prédéfinis, mais cette liberté permet d'utiliser cette extension quelle que soit la langue dans laquelle on écrit.
Enfin, j'ai cherché le bon compromis entre complexité du code et qualité des dessins obtenus avec cette extension : ils *ressemblent* à ceux du logiciel scratch, mais le but de cette extension n'est *pas* la ressemblance absolue au pixel près !

3 Les blocs normaux

Ces blocs sont les plus courants et possèdent une encoche d'emboitement, femelle en haut et mâle en bas. Les macros permettant de dessiner ces blocs ont des noms de la forme \block⟨*suffixe*⟩ et ont un seul argument obligatoire qui est le texte que l'on souhaite mettre dans le bloc. Par exemple, un bloc bleu (correspondant au menu « mouvement ») a un suffixe **move**, et est dessiné grâce à la macro \blockmove{⟨*texte*⟩}. Ainsi, dans l'environnement scratch, écrire \blockmove{Bonjour le monde} donne

La police d'écriture dans chaque bloc est la police « sans serif » en gras qui est définie dans le document au moment où l'environnement est appelé : pratiquement, cela signifie que les macros \sffamily et \bfseries sont exécutées avant que le texte des blocs ne soit composé. Dans cette documentation, la police sans serif est « biolinum ».
La plupart des dimensions des blocs sont proportionnelles à la taille de la police en cours. On peut donc jouer sur la taille de la police (via les classiques macros \small, \large, \footnotesize, etc) pour modifier la taille des blocs [3] .
Voici un inventaire des tous les blocs disponibles, empilés les uns sous les autres :

```
Voici un algorithme bizarre : \begin{scratch}
  \blockmove{bloc de mouvement}
  \blocklook{bloc d'apparence}
  \blocksound{bloc de son}
  \blockpen{bloc de stylo}
  \blockvariable{bloc de variable}
  \blocklist{bloc de liste}
  \blockevent{bloc d'événement}
  \blockcontrol{bloc de contrôle}
  \blocksensing{bloc de capteur}
\end{scratch}
```

2. Je ne déteste rien de plus comme langage de programmation que ce *truc* vaguement graphique, ultra limité et contre-productif dans l'apprentissage du codage qu'est scratch et que l'éducation nationale veut à tout prix imposer. Je ne compte bien évidemment pas me plier à ce nouveau dogme ridicule et ne l'utiliserai ni ne l'enseignerai ; je considère que les élèves méritent mieux que scratch – ADA par exemple – comme entrée dans le monde de la programmation. Ceci dit, coder cette extension a été un petit amusement.
3. Il y a aussi la clé « scale » pour mettre le dessin à l'échelle que l'on souhaite, voir page 8

Voici un algorithme bizarre :

Il faut donc retenir cette logique : les suffixes **move**, **look**, **sound**, **pen**, **variable**, **list**, **event**, **control** et **sensing** correspondent aux couleurs des blocs. Il existe aussi le suffixe **operator** qui n'a pas été montré précédemment puisqu'aucun bloc n'existe pour la fonction « opérateurs ».

4 Les ovales

Les « ovales » sont, selon le code graphique de `scratch`, censés contenir des nombres. Lorsque ces nombres sont explicitement écrits en chiffres, ces ovales ont un fond blanc :

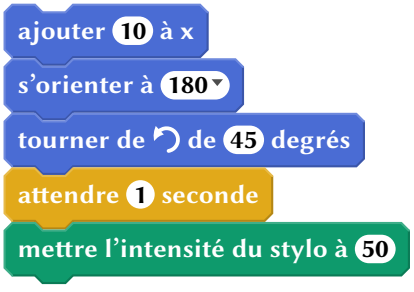
Les valeurs numériques sont parfois issues de valeurs prédéfinies auquel cas, une flèche de sélection doit apparaitre après le nombre prédéfini choisi :

Au niveau des macros utilisées dans l'exemple plus bas :

- la macro `\ovalnum{`⟨*nombre*⟩`}` dessine un nombre dans un ovale à fond blanc ;
- la macro `\selectarrownum` trace la flèche de sélection ;
- les macros `\turnleft` et `\turnright` dessinent des flèches de rotation dans les blocs `\blockmove`.

```
Ovales sur fond blanc : \begin{scratch}
  \blockmove{ajouter \ovalnum{10} à x}
  \blockmove{s'orienter à \ovalnum{180\selectarrownum}}
  \blockmove{tourner de \turnleft{} de \ovalnum{45} degrés}
  \blockcontrol{attendre \ovalnum{1} seconde}
  \blockpen{mettre l'intensité du stylo à \ovalnum{50}}
\end{scratch}
```

Ovales sur fond blanc :



Lorsque les ⟨*nombres*⟩ sont contenus dans des variables, les ovales prennent alors la couleur de la fonction correspondant à ces variables. Les macros ont des noms de la forme `\oval`⟨*suffixe*⟩ et ont pour suffixe **move**, **look**, **sound**, **variable**, **list**, **sensing** et **operator**.
Voici quelques exemples :

```
Ovales divers : \begin{scratch}
  \blockmove{s'orienter à \ovalmove{direction}}
  \blocksound{ajouter \ovaloperator{\ovalnum{-0.5} * \ovalsound{volume}} au volume}
  \blockmove{glisser en \ovalsound{tempo} secondes à x: \ovalsensing{souris x}
             y: \ovaloperator{\ovalnum{4} * \ovalsensing{souris y}}}
\end{scratch}

Toutes les macros ayant pour préfixe \verb|\oval| sont utilisables \emph{en dehors} de l'environnement
\texttt{scratch}, la preuve : voici \ovalvariable{une variable} et \ovaloperator{un opérateur}.
```

Ovales divers :

Toutes les macros ayant pour préfixe `\oval` sont utilisables *en dehors* de l'environnement `scratch`, la preuve : voici une variable et un opérateur.

5 Les blocs de début

Ces blocs sont de la couleur **event** pour la plupart (macro `\blockinit`), mais il existe aussi un bloc de début de couleur **control** (macro `\blockinitclone`). Le drapeau vert est dessiné avec la macro `\greenflag`.

```
Voici un début :
\begin{scratch}
  \blockinit{quand \greenflag est cliqué}
  \blockmove{suite de l'algorithme...}
\end{scratch}
et un autre :
\begin{scratch}
  \blockinitclone{quand je commence comme un clone}
  \blocklook{enfiler le costume de clown !}
\end{scratch}
```

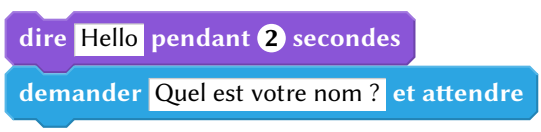
Voici un début :

et un autre :

6 Les rectangles

Du texte spécifié par l'utilisateur se trouve dans un rectangle sur fond blanc, en graisse normale et en couleur noir. La macro `\txtbox{⟨du texte⟩}` permet, dans l'environnement `scratch` d'insérer ces rectangles « de texte » :

```
\begin{scratch}
  \blocklook{dire \txtbox{Hello} pendant \ovalnum{2} secondes}
  \blocksensing{demander \txtbox{Quel est votre nom ?} et attendre}
\end{scratch}
```



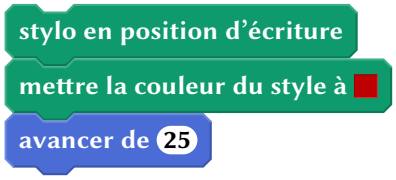
Un menu déroulant ayant des valeurs prédéfinies est également symbolisé par un rectangle dont la couleur reprend la fonction du bloc qui le contient. Pour ce faire, la macro `\selectmenu{⟨texte⟩}` doit être exécutée :

```
\begin{scratch}
  \blockinit{Quand je reçois \selectmenu{message 1}}
  \blockcontrol{créer un clone de \selectmenu{moi même}}
```

```
  \blockmove{aller à \selectmenu{pointeur de souris}}
  \blocklook{ajouter à l'effet \selectmenu{couleur} \ovalnum{25}}
\end{scratch}
```

Un carré rempli de couleur et accessible avec la macro \squarecolor{⟨*couleur*⟩} achève cet inventaire sur les rectangles :

```
\begin{scratch}
  \blockpen{stylo en position d'écriture}
  \blockpen{mettre la couleur du style à \squarecolor{red!75!black}}
  \blockmove{avancer de \ovalnum{25}}
\end{scratch}
```



7 Les losanges et les blocs de test

Dans la symbolique graphique de scratch, les losanges contiennent des valeurs booléennes ayant vocation à se retrouver dans un bloc de test. Pour dessiner de tels objets booléens, les macros \bool⟨*suffixe*⟩{⟨*texte*⟩} sont utilisées où les ⟨*suffixes*⟩ représentent les couleurs correspondant à la fonction du booléen tracé : **list**, **sensing** ou **operator**.
Les blocs de test sont de deux types,selon qu'ils possèdent ou pas une branche « else ».

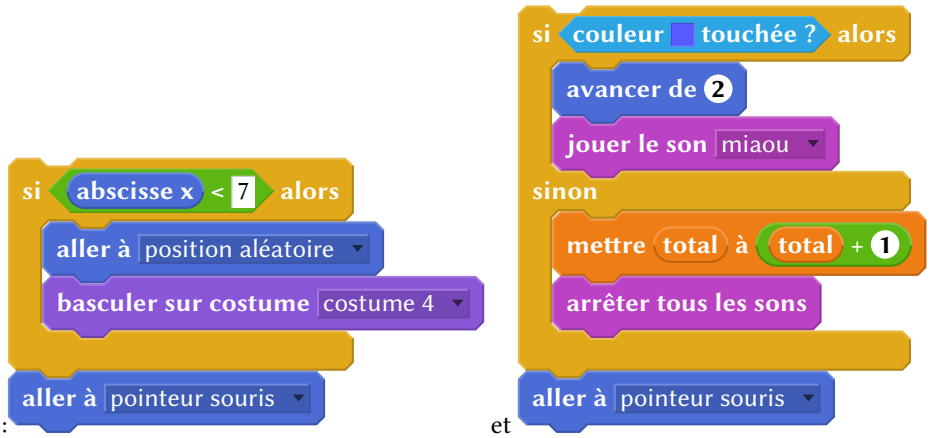
```
\blockif{⟨texte du test⟩}
    {⟨instructions si test vrai⟩}
```

et

```
\blockifelse{⟨texte du test⟩}
    {⟨instructions si test vrai⟩}
    {⟨instructions si test faux⟩}
```

```
Conditions :\begin{scratch}
  \blockif{si \booloperator{\ovalmove{abscisse x} < \txtbox{7}} alors}
    {\blockmove{aller à \selectmenu{position aléatoire}}
     \blocklook{basculer sur costume \selectmenu{costume 4}}
    }
  \blockmove{aller à \selectmenu{pointeur souris}}
\end{scratch}
et
\begin{scratch}
  \blockifelse{si \boolsensing{couleur \squarecolor{blue!65} touchée ?} alors}
    {\blockmove{avancer de \ovalnum{2}}
    \blocksound{jouer le son \selectmenu{miaou}}
    }
    {\blockvariable{mettre \ovalvariable{total} à \ovaloperator{\ovalvariable{total} + \ovalnum{1}}}
    \blocksound{arrêter tous les sons}
    }
  \blockmove{aller à \selectmenu{pointeur souris}}
```

```
\end{scratch}
```

Conditions : 

Les macros `\squarecolor` et celles de la forme `\bool`⟨*suffixe*⟩ sont utilisables *en dehors* d'un environnement `scratch` :

```
Une couleur : \squarecolor{cyan}.\par
Un booléen : \boollist{liste \selectmenu{malist} contient \txtbox{foobar}}.\par
Un autre : \booloperator{\booloperator{\ovalvariable{varx} > \txtbox{1}}
et \booloperator{\ovalvariable{varx} < \txtbox{5}}}.
```

Une couleur : .
Un booléen : liste malist contient foobar.
Un autre : varx > 1 et varx < 5.

8 Les blocs de fin

Ces blocs sont susceptibles de clore un algorithme et n'ont donc pas d'encoche mâle dans leur partie basse. Ils ne peuvent être que du type **control** et sont dessinés avec la macro `\blockstop{`⟨*texte*⟩`}`

```
\begin{scratch}\blockstop{supprimer ce clone}\end{scratch}
ou
\begin{scratch}\blockstop{stop \selectmenu{ce script}}\end{scratch}
```

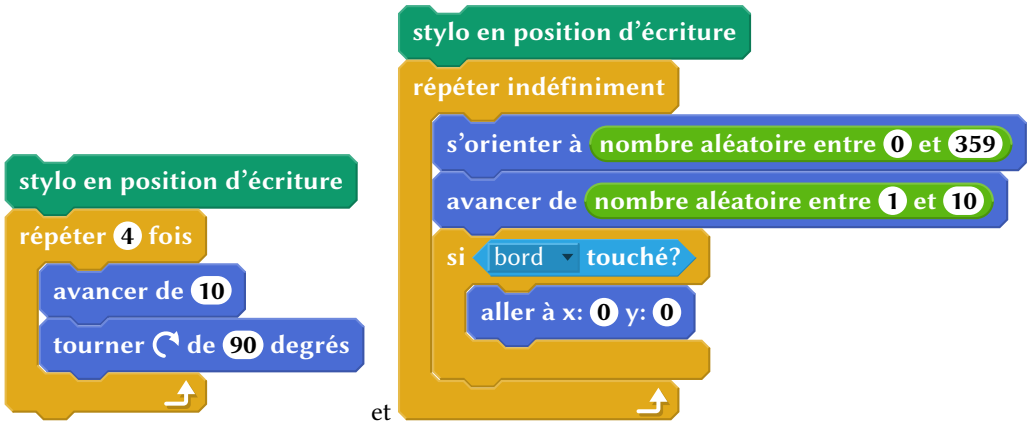
supprimer ce clone ou stop ce script

9 Les blocs de répétition

Ces blocs sont de deux types, selon que la répétition est prévue pour s'arrêter ou pas (boucle infinie). Ils seront dessinés par les macros `\blockrepeat` et `\blocinfloop` ayant chacune *deux* arguments : le premier étant le ⟨*texte*⟩ du bloc et le second la suite d'instructions à répéter.

```
\begin{scratch}
  \blockpen{stylo en position d'écriture}
  \blockrepeat{répéter \ovalnum{4} fois}
    {
    \blockmove{avancer de \ovalnum{10}}
    \blockmove{tourner \turnright{} de \ovalnum{90} degrés}
    }
\end{scratch}
et
\begin{scratch}
  \blockpen{stylo en position d'écriture}
  \blockinfloop{répéter indéfiniment}
    {
    \blockmove{s'orienter à \ovaloperator{nombre aléatoire entre \ovalnum{0} et \ovalnum{359}}}
    \blockmove{avancer de \ovaloperator{nombre aléatoire entre \ovalnum{1} et \ovalnum{10}}}
    \blockif{si \boolsensing{\selectmenu{bord} touché?}}
      {
```

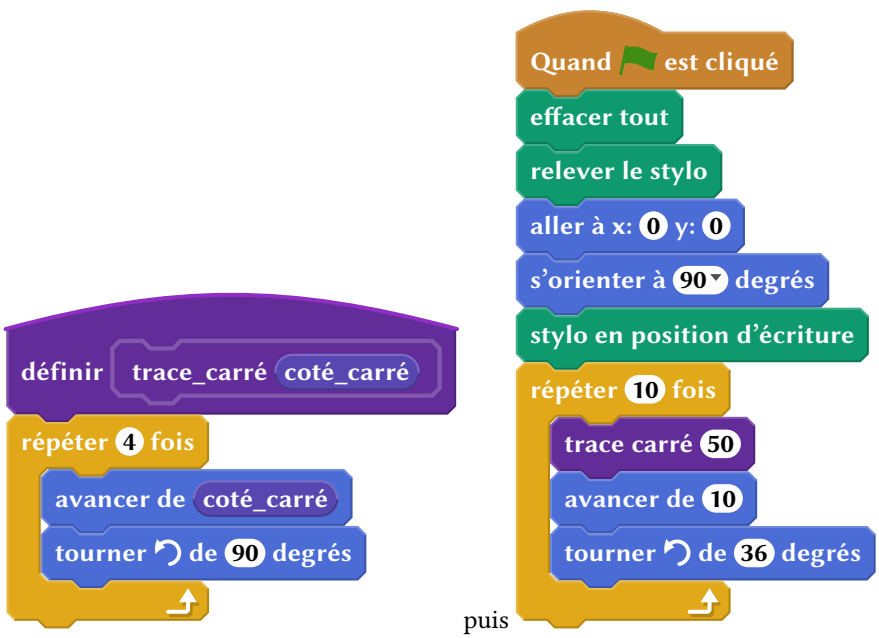
```
      \blockmove{aller à x: \ovalnum{0} y: \ovalnum{0}}
      }
    }
\end{scratch}
```



10 Les blocs de définition

Les « procédures », ayant le suffixe **moreblocks**, permettent d'étendre les maigres possibilités données au programmeur en scratch.

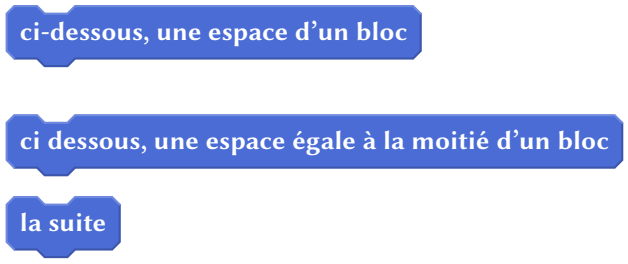
```
\begin{scratch}
  \initmoreblocks{définir \namemoreblocks{trace_carré \ovalmoreblocks{coté_carré}}}
  \blockrepeat{répéter \ovalnum4 fois}
    {\blockmove{avancer de \ovalmoreblocks{coté_carré}}
    \blockmove{tourner \turnleft{} de \ovalnum{90} degrés}
    }
\end{scratch}
puis
\begin{scratch}
  \blockinit{Quand \greenflag est cliqué}
  \blockpen{effacer tout}
  \blockpen{relever le stylo}
  \blockmove{aller à x: \ovalnum0 y: \ovalnum0}
  \blockmove{s'orienter à \ovalnum{90\selectarrownum} degrés}
  \blockpen{stylo en position d'écriture}
  \blockrepeat{répéter \ovalnum{10} fois}
    {
    \blockmoreblocks{trace carré \ovalnum{50}}
    \blockmove{avancer de \ovalnum{10}}
    \blockmove{tourner \turnleft{} de \ovalnum{36} degrés}
    }
\end{scratch}
```



11 Bloc invisible

Bien que ce genre de bloc n'existe pas avec `scratch`, cette fonctionnalité peut s'avérer utile. On insère un espace vide avec `\blockspace[`⟨*coeff*⟩`]`. L'espace verticale insérée est égale à la hauteur normale d'un bloc multipliée par le ⟨*coeff*⟩, valeur optionnelle qui vaut 1 par défaut.

```
\begin{scratch}
  \blockmove{ci-dessous, une espace d'un bloc}
  \blockspace
  \blockmove{ci dessous, une espace égale à la moitié d'un bloc}
  \blockspace[0.5]
  \blockmove{la suite}
\end{scratch}
```



12 Personnalisation

Plusieurs ⟨*paramètres*⟩ peuvent être réglés par l'utilisateur selon la syntaxe ⟨*clé*⟩=⟨*valeur*⟩. Cas paramètres peuvent être spécifiés dans :

- l'argument optionnel de l'environnement `\begin{scratch}[`⟨*paramètres*⟩`]` auquel cas les ⟨*paramètres*⟩ ne s'appliquent qu'à cet environnement ;
- l'argument de la macro `\setscratch{`⟨*paramètres*⟩`}` pour spécifier des ⟨*paramètres*⟩ pour les environnements `scratch` à venir ;
- l'argument de `\setdefaultscratch{`⟨*paramètres*⟩`}` pour spécifier des ⟨*paramètres*⟩ *par défaut*.

Il existe la macro `\resetscratch` qui remet à leur valeur par défaut tous les ⟨*paramètres*⟩ de `scratch`, pour annuler les effets d'une macro `\setscratch`.

Voici les ⟨*paramètres*⟩ disponibles :

else word=⟨caractères⟩ (*Défaut* : sinon)
Représente est le mot qui est inséré dans la branche « else » d'un bloc de test.

x sep=⟨dimension⟩ (*Défaut* : 0.5em)
Représente l'espace horizontale insérée entre les bords droit et gauche du texte du bloc et les bords droits et gauche du bloc.

y sepsup=⟨dimension⟩ (*Défaut* : 1pt)
Représente l'espace verticale insérée entre le bas de l'encoche femelle et le bord supérieur du texte du bloc.

y sepinf=⟨dimension⟩ (*Défaut* : 3pt)
Représente l'espace verticale insérée entre le bas du bloc et le bord inférieur du texte du bloc.

line width=⟨dimension⟩ (*Défaut* : 0.8pt)
Représente l'épaisseur des lignes de relief des blocs et le double des lignes de relief des losanges booléens.

loop width=⟨dimension⟩ (*Défaut* : 3ex)
Représente est la largeur de la barre verticale des blocs de répétition ou de test.

loop height=⟨dimension⟩ (*Défaut* : 1.75ex)
Représente est l'épaisseur des barres horizontales « else » et inférieure des blocs de répétition ou de test.

corner=⟨dimension⟩ (*Défaut* : 0.66667ex)
Représente la dimension des chanfreins des blocs. Cette dimension est utilisée proportionnellement pour le placement horizontal, l'épaisseur et la largeur des encoches des blocs.

scale=⟨coefficient⟩ (*Défaut* : 1)
Représente l'échelle à laquelle est représenté le dessin.

init arcangle=⟨angle⟩ (*Défaut* : 30)
Représente l'angle avec l'horizontale de l'arc de cercle tracé dans la partie haute des blocs de départ.

init arclength=⟨dimension⟩ (*Défaut* : 5em)
Représente la longueur horizontale de l'arc de cercle tracé dans la partie haute des blocs de départ.

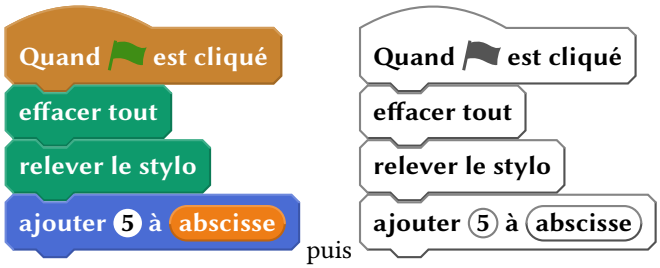
moreblock arcangle=⟨angle⟩ (*Défaut* : 15)
Représente l'angle avec l'horizontale de l'arc de cercle tracé dans la partie haute des blocs de de type « moreblock ».

contrast=⟨entier⟩ (*Défaut* : 20)
Cet entier, compris entre 0 et 100 inclus, qualifie la différence de teintes entre les lignes de relief tracées autour des blocs. L'entier 0 signifie « aucun contraste » (teintes identiques) tandis que que 100 signifie « contraste maximal » auquel cas les lignes claires, qui se situent sur la partie haute des blocs, sont blanches et les lignes foncées sont noires.

print=⟨booléen⟩ (*Défaut* : false)
Lorsque ce booléen est vrai, les dessins se font en noir et blanc de façon à pouvoir être dirigés vers une impression en noir et blanc.

```
\begin{scratch}
  \blockinit{Quand \greenflag est cliqué}
  \blockpen{effacer tout}
  \blockpen{relever le stylo}
  \blockmove{ajouter \ovalnum{5} à \ovalvariable{abscisse}}
\end{scratch} puis
\begin{scratch}[print]
  \blockinit{Quand \greenflag est cliqué}
  \blockpen{effacer tout}
  \blockpen{relever le stylo}
  \blockmove{ajouter \ovalnum{5} à \ovalvariable{abscisse}}
```

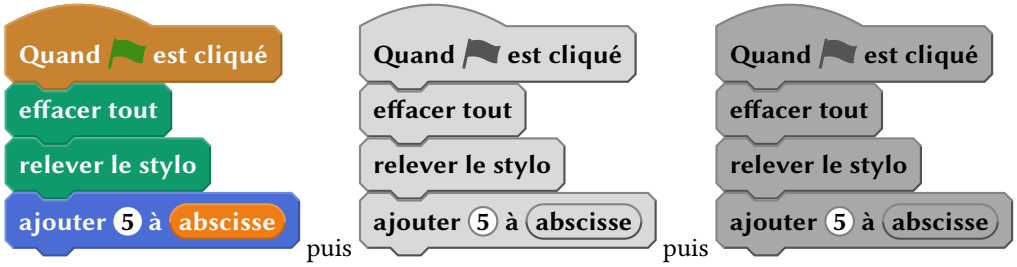
```
\end{scratch}
```



fill blocks=⟨booléen⟩ (*Défaut* : `false`)
Ce booléen n'est pris en compte que lorsque le booléen `print` est vrai. Si `fill blocks` est vrai, tous les dessins (sauf les ovales contenant des nombres) seront remplis avec un gris choisi avec la clé suivante.

fill gray=⟨taux de gris⟩ (*Défaut* : `0.85`)
Lorsque `fill blocks` est vrai, ce taux de blanc dans le gris (nombre compris entre 0 pour noir et 1 pour blanc) est utilisé pour définir une couleur de remplissage des dessins.

```
\begin{scratch}
  \blockinit{Quand \greenflag est cliqué}
  \blockpen{effacer tout}
  \blockpen{relever le stylo}
  \blockmove{ajouter \ovalnum{5} à \ovalvariable{abscisse}}
\end{scratch} puis
\begin{scratch}[print,fill blocks]
  \blockinit{Quand \greenflag est cliqué}
  \blockpen{effacer tout}
  \blockpen{relever le stylo}
  \blockmove{ajouter \ovalnum{5} à \ovalvariable{abscisse}}
\end{scratch} puis
\begin{scratch}[print,fill blocks,fill gray=0.66]
  \blockinit{Quand \greenflag est cliqué}
  \blockpen{effacer tout}
  \blockpen{relever le stylo}
  \blockmove{ajouter \ovalnum{5} à \ovalvariable{abscisse}}
\end{scratch}
```

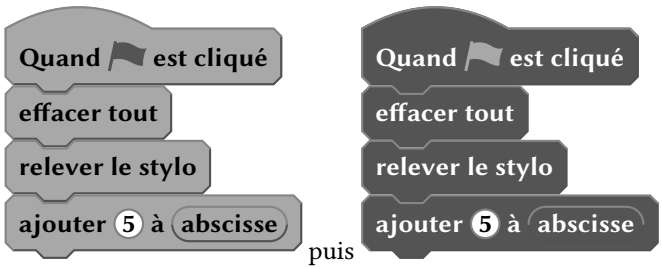


text color=⟨couleur⟩ (*Défaut* : `black`)
Lorsque `fill blocks` est vrai, cette couleur sera utilisée pour le texte des blocs.

flag gray=⟨taux de gris⟩ (*Défaut* : `0.33`)
Lorsque `print` est vrai, ce taux de gris est utilisé pour la couleur du drapeau tracé avec `\greenflag` ainsi que pour la flèche se trouvant au bas des blocs de répétition.

```
\begin{scratch}[print,fill blocks,fill gray=0.66]
  \blockinit{Quand \greenflag est cliqué}
  \blockpen{effacer tout}
  \blockpen{relever le stylo}
  \blockmove{ajouter \ovalnum{5} à \ovalvariable{abscisse}}
\end{scratch} puis
\begin{scratch}[print,fill blocks,fill gray=0.33,text color=white,flag gray=0.66]
  \blockinit{Quand \greenflag est cliqué}
```

```
  \blockpen{effacer tout}
  \blockpen{relever le stylo}
  \blockmove{ajouter \ovalnum{5} à \ovalvariable{abscisse}}
\end{scratch}
```



line gray=⟨taux de gris⟩ (*Défaut* : 0.4)
Lorsque print est vrai, ce taux de gris est utilisé pour la couleur des lignes de relief.

13 Mot de la fin

Le code de cette extension démontre mon immense ignorance de ti*k*z/pgf et les méthodes de programmation qui lui sont propres que, décidément, je ne comprendrai jamais ! C'est sans doute le trop grand éloignement avec la logique de TEX et la documentation de ti*k*z/pgf, aussi indigeste qu'illisible, qui explique cette incompatibilité d'humeur et mon désintérêt à l'égard de ti*k*z. Toujours est-il que cette extension fonctionne, avec une lenteur certaine que j'attribue à ma programmation hasardeuse ainsi qu'à la lenteur intrinsèque de ti*k*z.
Toute remarque, remontée de bug — je n'ose pas dire amélioration du code —, demande d'implémentation de fonctionnalité est bien évidemment bienvenue ; j'invite les utilisateurs à m'en faire part *via* email à unbonpetit@netc.fr

14 Historique

v0.1 16/05/2017 Première version.

v0.2 28/05/2017 Pour une impression en noir et blanc, ajout de l'option « print », suite à une demande de P. Celdran. De cette option découlent les options « fill blocks », « fill gray », « text color », « flag gray » et « line gray ». Ajout également de l'option « contrast » qui qualifie la différence de nuance entre les lignes de contraste des parties hautes et des parties basses.

v0.21 24/07/2017 Correction d'une erreur dans le tracé des lignes de contraste des "boolbox".
Correction d'un bug dans \resetscratch.
Correction d'un bug dans \txtbox.
Chargement de l'extension simplekv pour les clés/valeurs.
Ajout de la clé « scale ».